НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему Апаратно-програмний комплекс моніторингу та управління станом дорожнього покриття Smart City

Виконав: студент 4 курсу, групи ТВ-61

Шматко Андрій Іванович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Ковальчук А. М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н., ст. вик. кафедри ТЕУ Т і АЕС Сірий О.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Шматко Андрій Іванович
(прізвище, ім’я, по батькові)

1. Тема роботи Апаратно-програмний комплекс моніторингу та управління станом дорожнього покриття Smart City

керівник роботи доцент, к.т.н. Ковальчук Артем Михайлович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № 1168-с

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи мови прорамування C, PHP, Javascript, середовища розробки ArduinoIDE, PhpStorm, СУБД MySQL

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні решення для моніторингу та управляння станом дорожнього покриття , можливі методи реалізації взаємодії апаратних та програмнії модулів, обгрунтувати засоби розробки. Розробити систему для моніторингу та управління станом дорожнього покриття. Зробити висновки за результатами роботи.

5. Перелік ілюстративного матеріалу Архітектура багатоагентних систем. Функції системи. Схема агенту моніторингу. Архітектура програмної частини комплексу. Приклади роботи клієнтського додатку. _____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	11.10.2019 — 23.01.2020	
3.	Розробка архітектури та загальної структури системи	03.02.2020 — 04.03.2020	
4.	Розробка структур окремих підсистем	05.03.2020 — 12.04.2020	
5.	Програмна реалізація системи	13.04.2020 — 17.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020 — 07.06.2020	
7.	Захист програмного продукту	08.06.2020	
8.	Передзахист	08.06.2020	
9.	Захист	15.06.2020	

Студент _____ ___Шматко А. І._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ ___Ковальчук А. М._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Пояснювальна записка містить 63 сторінок, включає 20 рисунків, 3 таблиці, 3 додатки та 18 посилань.

Метою роботи є створення агенту для втконання задачі моніторингу метеоумов на дорожньому покритті, а також програмного агенту управління та інтерфейсу до нього у вигляді клієнтського додатку.

Результатом проботи є програмний агент моніторингу метеоумов на дорогах,, що був розроблений з використанням мови С, а також агент управління з інтерфейсом у вигляді клієнтського веь-застосунку, котрий було розробено на PHP та Javascript.

Ключові слова: програмний агент, система моніторингу та управління, багатоагентні системи, метеоумови на дорогах.

ABSTRACT

The explanatory note contains 63 pages, includes 20 figures, 3 tables, 3 appendices and 18 references.

The purpose of the work is to create an agent for fulfilling the task of monitoring meteorological conditions on the road surface, as well as a software management agent and an interface to it in the form of a client application.

The result of the work is a software agent for monitoring road weather conditions, which was developed using the C language, as well as a control agent with an interface in the form of a client-side application, which was developed in PHP and Javascript.

Keywords: software agent, monitoring and management system, multi-agent systems, weather conditions on the roads.

ЗМІСТ

ВСТУП

Проблема утримання автомобільних доріг в належному стані є досить актуальною, особливо в осінньо-зимовий період. Інтелектуальний моніторинг стану доріг на предмет метеорологічних факторів дозволяє своєчасно реагувати дорожнім службам та попереджати негативний вплив метеоумов на надійність функціонування доріг.

Інформаційні системи дорожньої погоди використовуються для спостережень за метеорологічними умовами на дорогах, прийняття рішень на основі зібраних даних, а також для розробки прогнозів та для відображення чи поширення інформації про погоду на дорозі.

Дані системи використовуються операторами дорожнього руху та обслуговуючими службами для ефективного прийняття рішень, особливо при несприятливих погодних умовах. Інформаційні системи дорожньої погоди дуже актуальні в аеропортах, на мостах, на ділянках дороги, розташованих біля обривів. Також на дорогах, що розташовані в долинах та на пагорбах, адже на них погодні умови можуть суттєво відрізнятися, а тому прогнози, зроблені гідрометцентром, не завжди будуть точними. У такому випадку доцільно прогнозувати наближення несприятливих погодних умов для конкретної місцевості.

Більшість доріг в осінньо-зимовий період знаходяться в зоні впливу мінусових температур. У цей період питання безпеки дорожнього руху безпосередньо пов'язані з вирішенням проблем оперативного видалення снігу і льодоутворень з дорожніх покриттів. Системи що дозволяють вчасно реагувати на несприятливі дорожні умови є затребуваними.

В даній роботі було запропоновано створити апаратно-програмний комплекс - інформаційну систему дорожньої погоди з можливостями моніторингу стану поверхні дорожнього покриття, прогнозування метеоумов та реагування на несприятливі умови. Було поставлено завдання: проаналізувати існуючі системи дорожньої погоди, та, врахувавши їх переваги і недоліки, розробити програмну та апаратну частини власної системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття багатоагентної системи

Агенти, а точніше інтелектуальні агенти - нова парадигма для розробки програмних систем, що підтримує моделювання складних індивідуальних взаємодій. Агентно-орієнтовані системи поєднують об’єктно-орієнтовану технологію програмування з технологіями штучного інтелекту. Поширення використання агентної парадигми в ІТ призводить до якісно нового рівня взаємодії користувача з ПЗ.

Багатоагентна система [1] — це система, що складається з декількох інтелектуальних агентів, що взаємодіють між собою. Багатоагентні системи - це потужний інструмент за допомогою якого можна вирішувати завдання, котрі неможливо або невигідно виконувати за допомогою одного агента. Прикладами таких завдань є системи реагування та ліквідації екстрених ситуацій, онлайн-торгівля, системи електронного документообігу, різні експертні системи та ін.

У багатоагентній системі агенти мають наступні властивості [2]:

— реактивність: здатність сприймати стан середовища і своєчасно реагувати на зміни в заданому середовищі;

—децентралізація: немає агентів, що керують усією системою;

—самостійність: агенти здатні функціонувати без прямого втручання людей і мають певний контроль над своїми діями та внутрішнім станом;

—соціальність: здатність взаємодіяти з іншими агентами і обмінюватись з ними повідомленнями з допомогою деякої мови комунікації;

—проактивність: агенти не просто діють у відповідь на своє оточення, вони здатні проявляти цілеспрямовану поведінку, беручи на себе ініціативу;

—обмеженість уявлення: у жодного з агентів немає повного уявлення про всю систему.

На рисунку 1.1 зображено приклад багатоагентної системи.

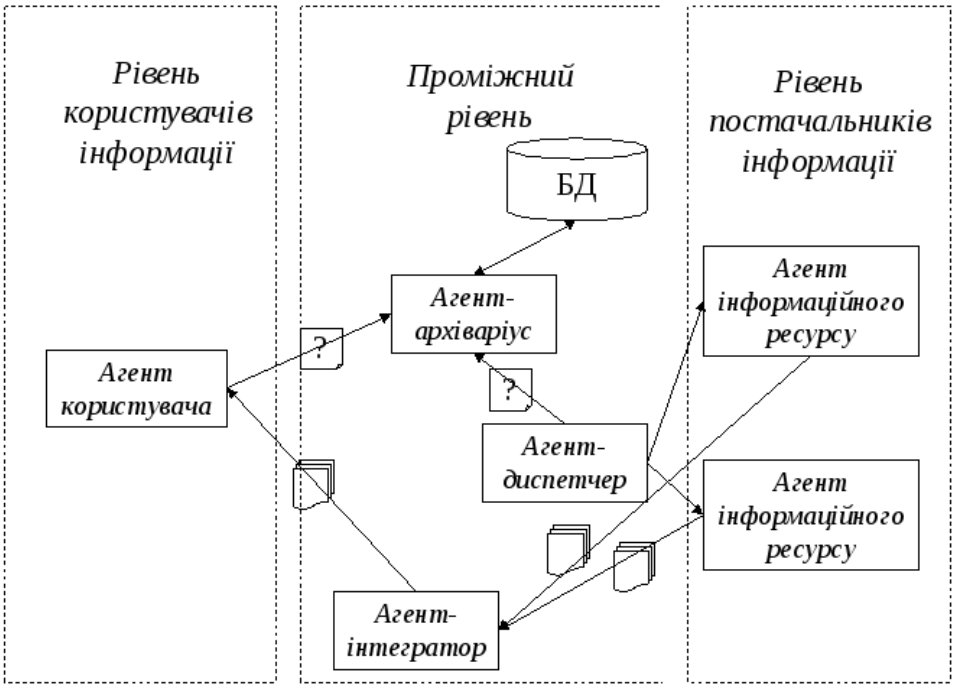


Рисунок 1.1 — Багатоагентна інформаційно-пошукова система

Агенти з моніторингу та спостереження

Дані агенти використовуються для спостереження за об'єктами та передачі інформації на комп'ютерні системи [3]. Вони здатні стежити за рівнем запасів матеріалів компанії, за цінами конкурентів і реагувати на їх зміни, і т.д.

Прикладом агенту моніторингу є агент в лабораторії реактивного руху NASA, який займається відстеженням наявності і стану обладнання та продуктів харчування. Також він здатний планувати замовлення на придбання нового обладнання з урахуванням оптимізації витрат. Такі агенти можуть здійснювати моніторинг складних комп'ютерних мереж і стежити за конфігурацією кожного комп'ютера, підключеного до мережі.

Агенти моніторингу можуть використовуватись для моніторингу стану активів (засобів транспортування, боєприпасів та ін.). Якщо дані агенти мають ціль та здатні приймати рішення, то вони намагаються досягти цілей з наявними активами, зводячи до мінімуму витрати активів при максимальному досягненні мети.

У даній роботі було вирішено використовувати агенти моніторингу для спостереження за метеорологічними показниками на дорогах. Агент збирає інформацію з давачів та передає її на агент контролю для збереження та подальшої обробки.

1.2 Огляд існуючих рішень

Агент моніторингу метеорологічних умов - це система, що виконує моніторинг метеорологічних показників на певній ділянці. При дослідженні існуючих рішень було виявлено три системи, які вирішують задачу моніторингу метеорологічних умов на дорогах.

Theweathernetwork.com - сервіс що надає інформацію про погодні умови на дорогах Канади. Перевага даного сервісу - система надає можливість зручно переглядати інформацію по обраній ділянці дороги. Недоліки - відсутня можливість подивитися динаміку зміни погодних характеристик.

Weather.gov - портал що надає функцію моніторингу доріг США в тому числі і метеорологічних умов на них. Переваги даного сервісу - наявні попередження про появу несприятливих погодних умов в майбутньому. Недоліки - незручний інтерфейс, карта, відсутня можливість подивитися динаміку зміни метеоумов.

Rwis.indot.in.gov - інформаційний ресурс, який дозволяє користувачам спостерігати за метеорологічними показниками на дорогах США. Перевагою даного ресурсу є зручний інтерфейс. Також, ресурс на відміну від попередніх надає можливість подивитися зміни метеоумов у вигляді таблиць. Недоліки - відсутність висновків по погоді на основі погодних показників.

1.3 Постановка задачі

На основі аналізу існуючих рішень була сформульована задача даної роботи, а також були визначені вимоги до розроблюваної системи.

Задачею даної роботи є розробка програмних агентів - агента моніторингу та агента контролю.

Проаналізувавши архітектуру системи та функції агента в ній, було сформульовано головні вимоги до створюваного агента:

— програмний агент моніторингу повинен отримувати показники про стан дороги з метеорологічних давачів, що входять до складу агента;

— агент повинен мати спосіб зв’язку з іншими агентами системи, зокрема з агентом контролю;

— агент повинен відправляти зібрані з давачів дані на подальшу обробку до програмного агента контролю;

— агент контролю, в свою чергу, повинен зберігати отримані дані, аналізувати їх;

— агент контролю повинен виконувати необхідні прогнози метеоумов на основі отриманих даних.

Було вирішено розробити клієнтський додаток, що надає графічний інтерфейс до агенту контролю. Додаток має надавати наступні можливості:

— система повинна мати інтерфейс перегляду інформації про стан метеорологічних умов на дорогах;

— повинен бути реалізований механізм попереджень та/або інших реагувань при наявності несприятливих погодніх умов;

— повинна бути можливість перегляду динаміки зміни метеоумов на графіку.

2. АРХІТЕКТУРА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Архітектура багатоагентної системи

На рисунку 2.1 зображена схема розроблюваної багатоагентної системи. Система складається з багатьох агентів моніторингу та одного агента контролю, котрий має зручний інтерфейс у вигляді клієнтського додатку.

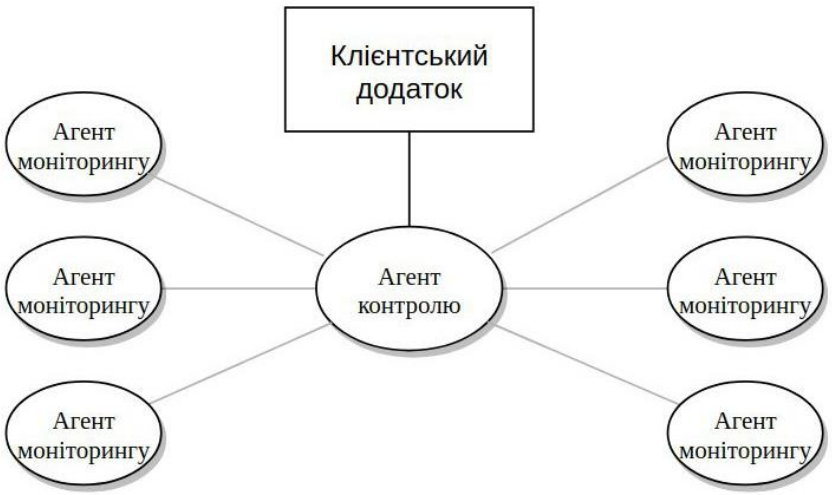


Рисунок 2.1 — Схема багатоагентної системи

2.2 Архітектура апаратної частини системи

2.2.1 Структурна схема пристрою

На рисунку 2.2 зображена схема агента моніторингу.

Агент моніторингу побудований на базі мікроконтролера. Складається з різноманітних давачів, що вимірюють метеорологічні показники. Агентам моніторингу потрібно комунікувати з агентом контролю та відсилати йому зібрані дані. Для забезпечення взаємодії між агентами, до агенту моніторингу повинен входити модуль комунікації.

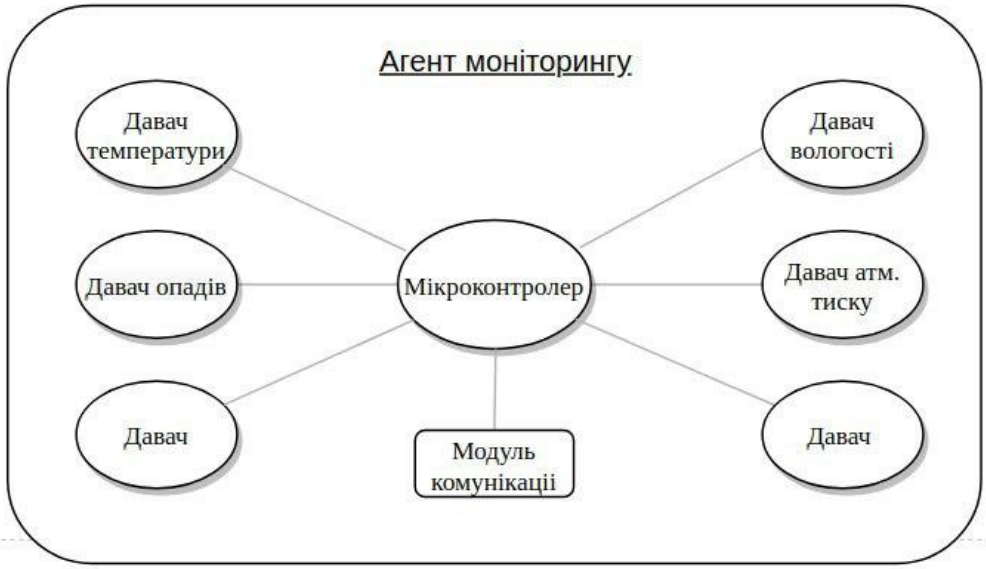


Рисунок 2.2 —Схема агента моніторингу

2.2.2 Вибір мікроконтролера

Arduino

Arduino - це комбінація апаратної і програмної частин для простої розробки електроніки. Апаратна частина включає в себе велику кількість видів плат Arduino з вбудованими програмованими мікроконтролерами, а також додаткові модулі. Програмна частина складається з середовища розробки (програми для написання ескізів і прошивки мікроконтролерів Arduino), спрощеної мови програмування, величезної кількості готових функцій і бібліотек.

Arduino - це платформа для зручної та швидкої розробки електронних пристроїв [4]. Платформа є популярною в усьому світі завдяки зручності і простоті мови програмування, а також відкритості архітектури і програмного коду. Мікроконтролер Arduino програмується через USB роз’єм без використання программаторів [5].

Прилади на базі Arduino можуть отримувати інформацію про навколишнє середовище за допомогою різних сенсорів, а також можуть управляти різними виконавчими пристроями.

З технічної точки зору, Arduino вміє приймати і відправляти сигнали відповідно до інструкцій в прошивці. Звучить досить скромно, але на практиці це дозволяє отримувати і обробляти інформацію з сенсорів і передавати команди виконавчим

механізмам або іншим пристроям. Наприклад, мікроконтролер може отримувати дані з датчиків температури, тиску, вологості і виводити отриману інформацію на дисплей.

Цих можливостей вистачає для реалізації складних пристроїв, таких як безпілотні літальні апарати, 3D-принтери, роботизовані маніпулятори, радіокеровані машинки, човни, всюдиходи та т. Д. Можливості Arduino обмежені тільки уявою. Якщо вам буде не вистачати можливостей Arduino, то існують більш потужні мікроконтролери такі як Arduino Mega, NodeMCU, STM32, Wemos, Raspberry Pi, Orange Pi [6].

Мікроконтролер на платі програмується за допомогою мови Arduino (заснований на мові Wiring) і середовища розробки Arduino IDE (заснована на середовищі Processing). Проекти пристроїв, засновані на Arduino, можуть працювати самостійно, або ж взаємодіяти з програмним забезпеченням на комп'ютері.

Далі буде розглянуто мікроконтролер сімейства Arduino, Arduino Mega (рисунок 2.3), який було обрано для реалізації агента. Буде виконано порівняння даного мікроконтролера з поширеним Arduino Uno.

Arduino Mega

Рисунок 2.3 — Мікроконтролер Arduino Mega

Arduino Mega побудований на базі мікроконтролера ATmega2560. Характеристики плати наведено у таблиці. Подати живлення на плату можна підключивши її до комп'ютера через USB кабель. Або ж подати живлення за допомогою адаптера AC / DC, або акумуляторною батареєю. Arduino Mega 2560 сумісна з усіма платами розширення, які розроблені для платформ Uno або Duemilanove.

Порівняємо Arduino Mega з популярним Arduino Uno (таблиця 2.1).

Таблиця 2.1. Порівняльна таблиця характеристик Arduino Uno та Arduino Mega

Характеристики	Arduino Uno	Arduino Mega
Мікроконтролер	ATmega328	ATmega2560
Робоча напруга	5В	5В
Вхідна напруга (рекомендована)	7-12В	7-12В
Вхідна напруга (гранична)	6-20В	6-20В
Цифрові входи / виходи	14 (6 з яких можуть працювати також як виходи ШІМ)	54 (14 з яких можуть працювати також як виходи ШІМ)
Аналогові входи	6	16
Постійний струм через вхід / вихід	40 mA	40 mA
Постійний струм для виведення	3.3 В 50 mA	3.3 В 50 mA

Флеш-пам'ять	32 КВ (з яких 0.5 КВ використовуються для завантажувача)	256 КВ (з яких 8 КВ використовуються для завантажувача)
ОЗУ	2 КВ	8 КВ
Енергонезалежна пам'ять	1 КВ	4 КВ
Тактова частота	16 МГц	16 МГц

Так як давачів у складі агента може бути багато і для них необхідно використовувати відповідні бібліотеки та завантажувати їх на мікроконтролер, то пам'яті на Arduino Uno не вистачить для даних цілей. Також потрібна чимала кількість пінів для підключення давачів та модулю комунікації. Отже, можливостей Arduino Uno не вистачить для виконання поставлених цілей і тому для реалізації агенту було обрано мікроконтролер Arduino Mega.

2.2.3 Вибір давача вологості, температури і атмосферного тиску

Найбільш оптимальним варіантом виявився барометр / термометр / давач вологості на чіпі BME280 (рисунок 2.4).

Рисунок 2.4 — Давач на чіпі BME280

Модуль на чіпі BME280 поєднує в собі датчик атмосферного тиску, термометр і датчик вологості повітря. В основі плати лежить чіп Bosch BME280, який є повним аналогом BMP280, але при цьому має в корпусі ще й датчик вологості повітря.

Датчик BME280 відрізняється від попередніх поколінь цифрових датчиків тиску - BMP085 і BMP180 зменшеними габаритами і енергоспоживанням, можливістю вимірювання вологості, підтримкою двох інтерфейсів зв'язку - SPI і I2C, великою роздільною здатністю показань по тиску і температурі, а також наявністю вбудованих фільтрів, які зменшують вплив шумів на свідчення датчиків. Властивості давача описані в таблиці 2.2.

Таблиця 2.2. Характеристики давача

Діапазон вимірювання	300...1100 hPa (-500...9000 м над рівнем моря)
Точність (тиск)	1 hPa
Напруга живлення	1.8 - 3.6 В
Струм	2.7 мкА (при частоті опитування 1 Гц)
Тип показань	цифровий вихід I2C
Робочий температурний діапазон	-40°C..+85°C
Розміри	15x11 мм

2.2.4 Вибір давача опадів

Рисунок 2.5 — Давач опадів

Було обрано простий давач води/дощу/снігу , що реагує на потрапляння дощу, води або снігу на контактну площадку (рисунок 2.5).

У модулі є два виходи: дискретний - видає лог "0" при перевищенні певного рівня вологості, що настроюється потенціометром, і аналоговий - видає аналогове значення пропорційно вологості.

Даний тип датчиків зазвичай використовують в метеостанціях на Arduino, системах автоматичного закриття вікон після початку дощу, сигналізаторах протікання і т.д.

2.2.5 Вибір модуля передачі

У багатьох додатках необхідно оперативно передавати інформацію про стан об'єкта моніторингу по бездротових каналах зв'язку. Одне з найбільш поширених та зручних для цього коштів - GSM-модуль, який вбудовується в датчики, розміщені на об'єкті.

Типовий GSM-модуль складається з радіоблоків (приймач, підсилювач та зовнішній радіочастотний інтерфейс), процесора, пам'яті і ряду інтерфейсів для інтеграції в кінцеві пристрої. GSM-модулі можуть передавати і приймати дані по каналах GSM і GPRS, в тому числі SMS-повідомлення і факси. Деякі модулі оснащені також GPS-приймачами і можуть визначати, а потім передавати координати об'єкта, в якому вони знаходяться. Більшістю GSM-модулів можна керувати за допомогою AT-команд.

Деякі виробники пропонують розширений набір цих команд, а також можливість програмування модулів на мовах високого рівня (наприклад, C і Python).

Одне з основних застосувань GSM модулів - в так званих системах M2M (Machine-to-Machine - міжмашинної комунікації). У них відбувається автоматичний обмін інформацією між різними системами збору і обробки даних. Для прийому / передачі цих даних і використовуються, в тому числі, GSM-модулі. Системи M2M можуть ефективно застосовуватися в багатьох областях. Ось кілька прикладів.

Медичні датчики з GSM-модулями дозволяють дистанційно стежити за здоров'ям людини. Дані з датчиків, встановлених на тілі пацієнта, передаються через мережу мобільного зв'язку на спеціальний портал. Лікарі або родичі можуть заходити

через Інтернет на цей портал і отримувати інформацію про стан пацієнта. У разі виникнення небезпечної ситуації можна зв'язатися з ним по тих же каналах (наприклад, надіславши SMS із запитом) і при необхідності викликати медичну допомогу. GSM-модулі вбудовуються і в стаціонарні датчики, які використовуються в M2M-системах контролю енергоспоживання, пожежної безпеки, охорони будівель і ін.

У M2M-системах управління транспортом на автомобілях встановлюють пристрої з GSM-модулями і GPS-приймачами, які через мережі мобільних операторів передають в спеціальні центри управління інформацію про координати і стан транспортних засобів, а при необхідності приймають з центру різні команди. Таким чином можна відстежувати місцезнаходження транспортних засобів, автоматично оплачувати проїзд, а також оптимізувати використання парку автотранспорту (наприклад, поліпшити диспетчеризацію, збільшити економію палива та ін.). Цікавий приклад можливої транспортної M2M-системи призводить компанія Cinterion . Електромобіль з GSM модулем, інтегрованим з датчиком заряду акумуляторів, стоїть на економічному режимі зарядки. Власник отримує на телефон дані про стан заряду. Якщо виникає термінове справа, а електромобіль не встигає достатньо зарядитися до потрібного часу, то власник відправляє по телефону команду про перехід на прискорений режим заряду.

Одне з найбільш масштабних перспективних застосувань технологій M2M – система e-Call (Emergency Call - екстрений виклик), призначена для оповіщення про автомобільні аварії і оперативного надання допомоги постраждалим. Вона включає спеціальні датчики, оснащені GSM-модулями і GPS-приймачами, які встановлюються на автомобілях. У разі автомобільної аварії датчики автоматично передають дані про координати, стані подушок безпеки та іншу інформацію на номер екстрених служб 112 по GSM-каналах і на місце аварії висилається допомога.

Всі GSM-модулі компанії SIM підтримують управління за допомогою розширеного набору AT-команд, мають набір вбудованих аудіокодеків і оснащені різними інтерфейсами.

Таким чином, сьогодні ринок пропонує безліч GSM-модулів різної функціональності і габаритів, виконаних в різних корпусах. З них можна підібрати варіант, оптимальний для конкретного додатка.

Агент моніторингу повинен спілкуватися з агентом контролю, надсилати йому дані, зібрані з давачів, на подальшу обробку. Так як на базі агента контролю буде реалізовано клієнтський додаток на клієнт-серверній архітектурі, яка зазвичай використовує HTTP протокол, то комунікацію між агентами було вирішено здійснювати за допомогою HTTP протоколу через мережу Internet.

Агент може знаходитись на ділянці дороги, де відсутні будь-які мережі, що надають доступ в Internet. Однак в даних місцях повинне бути мобільне покриття. Тому для забезпечення комунікації між агентами було вирішено використовувати GSM мережу для виходу в Internet.

Плата на базі GSM / GPRS модуля SIM800L - простий засіб для підключення різних пристроїв до GSM мережі (рисунок 2.6).

Рисунок 2.6 — Плата на базі модуля SIM800L

Плата має всю необхідну обв'язку, а також вбудований роз'єм для карти micro-SIM. За допомогою даного модуля можна приймати і відправляти SMS, здійснювати дзвінки, відправляти дані через E-Mail і т.д. Характиристики плати наведено в таблиці 2.3.

,

Таблиця 2.3. Характеристики модулю.

Напруга	3.7...4.2 В
Діапазони, що підтримуються	GSM 850/ 900/ 1800/ 1900 МГц
Клас потужності	4 (2 Вт в діапазонах 850/ 900 МГц)
Клас потужності	1 (1 Вт в діапазонах 1800/1900MHz)
Розмір	23* 25 * 7 мм

Управляти модулем можна за допомогою персонального комп'ютера через перетворювач інтерфейсу USB-UART або безпосередньо через UART модуль мікроконтролера Arduino, Raspberry Pi і аналогічних.

Компонент SIM800L має реалізований стек протоколу TCP / IP. Містить мікросхему MT6260SA компанії MediaTek і мікросхему приймача RFMD RF7176.

Завдяки функції відправки SMS повідомлень найбільш часто модуль GSM GPRS SIM800 MicroSIM з антеною використовується в диспетчеризації, бездротовій сигналізації і в охоронних системах [7]. При цьому в результаті різних подій відбувається відправлення повідомлень виду: "Аварійна зупинка ліфта 3 будинку №17", "Гараж відкритий", "Двері підвалу відкриті", "Протікання системи опалення", "Опалювальний котел вимкнений" , "Температура в теплиці нижче норми".

До модулю GSM GPRS SIM800 MicroSIM підключаються динамік і мікрофон. З модуля можна здійснювати і приймати дзвінки.

3. ЗАСОБИ РОЗРОБКИ

3.1 Arduino, C, C++

Мова програмування пристроїв Arduino заснована на C / C++ і скомпонована з бібліотекою AVR Libc, що дозволяє використовувати будь-які її функції. Разом з тим вона проста в освоєнні, і на даний момент Arduino - це, мабуть, найзручніший спосіб програмування пристроїв на мікроконтролерах.

C ++ - це одна з найпопулярніших мов, в основному використовується для написання системного та прикладного програмного забезпечення, драйверів, клієнтсько-серверних програмами та прошивок.

Основна особливість C++ - це сукупність заздалегідь визначених класів, які представляють собою типи даних, які можна інстанціювати кілька разів. Мова також полегшує декларування визначених користувачем класів. Класи можуть додатково вміщувати функції членів для реалізації певної функціональності.

Для реалізації функцій усередині класу можна визначити кілька об'єктів певного класу. Об'єкти можна визначити як екземпляри, створені під час виконання. Ці класи також можуть бути успадковані іншими новими класами, які за замовчуванням приймають загальнодоступні та захищені функції.

C ++ включає декілька операторів, таких як порівняння, арифметика, обробка бітів та логічні оператори. Однією з найпривабливіших особливостей C++ є те, що вона забезпечує перевантаження певних операторів, наприклад додавання.

Деякі з найважливіших концепцій мови програмування C++ включають поліморфізм, віртуальні функції та дружні функції, шаблони, простори імен та вказівники.

3.1.1 Arduino IDE

Для початку роботи з Arduino вам знадобиться спеціальне програмне забезпечення. Це середовище для розробки прошивок Arduino IDE. У цій програмі легко і зручно писати ескізи і завантажувати їх на ваш мікроконтролер Arduino. У середовищі розробки вже попередньо встановлено велику кількість і додаткових бібліотек.

Arduino IDE це офіційне програмне забезпечення з відкритим кодом, представлене Arduino.cc, яке призначене для редагування, компіляції та завантаження коду в Arduino плати. Практично всі модулі Arduino сумісні з цим програмним забезпеченням. Дане середовище розробки є відкритим кодом і його можна легко встановити та розпочати компілювати код.

Середовище доступне для таких операційних систем, як MacOS, Windows, Linux і працює на платформі Java, яка має вбудовані функції та команди, які відіграють життєво важливу роль для налагодження, редагування та компіляції коду в оточенні. Програму, написана в середовищі Arduino прийнято називати ескіз. Ескіз, створений на платформі IDE, в кінцевому підсумку генерує Hex файл, який потім передається та завантажується в контролер. Середовище IDE містить дві основні частини: редактор, який використовується для написання необхідного коду та компілятор який компілює та завантажує код у модуль Arduino. Середовище підтримує мови C і C++.

3.2 PHP

PHP є серверною мовою опису сценаріїв. Мова призначена для динамічного формування web сторінок та застосовується для створення web-застосунків [8].

PHP є відкритим і безкоштовним кодом. Більшість серверів веб-хостингу підтримують PHP за замовчуванням на відміну від інших мов, таких як C#, яким потрібен специфічний сервер IIS. Це робить PHP економічно вигідним вибором.

Мова програмування має вбудовану підтримку роботи з MySQL, тому досить просто організувати взаємодію додатку з базою даних. Мова є мультиплатформною, це означає є, що ви можете розгорнути свою програму в різних операційних системах, таких як Windows, Linux, Mac OS тощо.

Мову доцільно обирати для реалізації додатків, для яких не потрібна дуже висока швидкодія. PHP являється інтерпретованою мовою і має меншу швидкість виконання програми, ніж у разі використання компільованих мов. Якщо для розроблюваної системи важливі високі навантаження та висока продуктивність, тоді рекомендується обирати серед компільованих мов програмування. Коли для системи швидкодія не є критичним параметром, то варто обирати PHP. Зазвичай продуктивність PHP цілком достатня для створення серйозних WEB-додатків.

В PHP існують драйвери для роботи з популярними БД MySQL, MSSQL, PostgreSQL, Oracle.

Окрім створення динамічних веб сторінок, у PHP є інша область використання - це написання скриптів, що виконуються у командному рядку. Для такої роботи необхідний інтерпретатор PHP CLI, котрий працює незалежно від веб сервера. Цей спосіб роботи підходить, наприклад, для сценаріїв котрі повинні виконуватись систематично [9].

PHP має простий та зручний синтаксис. Мова підтримується майже на всіх відомих платформах, і майже на всіх операційних системах.

PHP - найбільш популярна в світі серверна скриптова мова програмування. Вона пройшла великий шлях розвитку від невеликих, вбудованих в код статичних HTML сторінок, фрагментів, до сучасної мови, на якій розробляється більшість сучасних динамічних сайтів.

3.2.1 PhpStorm

PhpStorm - це кросплатформне середовище розробки для PHP. Програма представляє багатофункціональний редактор для мов PHP, HTML, JavaScript та ін., а також має багато інших можливостей.

Дане середовище розробляється компанією JetBrains на базі продукту IntelliJ IDEA. IDE ідеально підходить для роботи з фреймоврками Symfony, Laravel, Yii, Drupal, Magento, Zend Framework, Joomla, CakePHP.

Редактор аналізує структуру коду та підтримує автодоповнення коду, рефакторинги, та запобігає помилкам.

Програмне середовище розробки також підтримує фронтенд технології. В ньому можна працювати з сучасними технологіями такими як HTML5, CSS, Sass, Less, Stylus, CoffeeScript, TypeScript, Emmet и JavaScript. При цьому доступні відладка та Unit-тестування.

IDE має підтримку систем контролю версій, підтримує віддалене розгортання, бази даних, інструменти командного рядка, Composer, Docker, REST-кліент та багато інших інструментів.

Редактор допомагає при написанні коду за допомогою інспекцій, що перевіряють код на льоту та аналізують весь проект цілком. Підтримка PHPDoc, інструментів форматування, автовиправлення помилок та інші можливості допомагають розробникам писати акуратний код, який легко підтримувати.

IDE надає зручні засоби навігації та пошуку по файлах та усьому проекту.

PhpStorm Community Edition випускається під ліцензією Apache, та доступний для всіх студентів, а також є Professional Edition з додатковими функціями – випущеними під власною ліцензією.

Основні можливості:

— Підсвічування коду;

— Багаті інструменти рефакторингу;

— Аналіз коду на льоту;

— Підтримка HTML5, JSDoc, Emmet;

— Підтримка Node.js;

— Віддалене розгортання по протоколах FTP, SFTP та ін.;

— Інтеграція з системами управління версіями: Git, GitHub що дозволяє оперативно оновлювати програмний код;

— Інтеграція з системами стеження за вадами коду;

— Відладка та налагодження коду;

— Автоматична підстановка закриваючих тегів, що значно прискорює процес написання коду;

— LiveEdit - редагування файлів «на льоту». Дозволяє одночасно редагувати код html, css або javascript і бачити, як результат змінюється в браузері. Для цього потрібна підтримка такої можливості з боку браузера, тому при установці необхідно ставити плагін для Google Chrome. Плагін працює з браузерами Google Chrome і його похідними;

— Засоби автоматизованого рефакторингу;

— Запобігання помилок в коді;

— Інтеграція з фреймворком тестування PHPUnit.

3.3 Laravel

Складність сучасних систем зростає і тому щоб кожного разу знову не писати код, що вирішує одну і ту ж задачу, були розроблені фреймворки. Фреймворки призначені для спрощення створення програмних додатків і полегшення процесу їх розробки.

Laravel - це фреймворк для web-додатків. Він має виразний синтаксис. Даний фреймворк дозволяє спростити та прискорити вирішення рутинних задач. Laravel надає засоби для зручної роботи з сесіями, аутентифікацією, базами даних, кешуванням, маршрутизацією, управлінням правами користувачів та ін. Фреймворк увібрав у себе все найкраще з інших PHP фреймворків.

До основних переваг Laravel належить:

— потужна екосистема, котра надає можливості розгортання застосунку на серверах;

— чудова документація з безліччю інформації;

— він має зручний механізм обробки помилок;

— фреймворк містить вбудовані засоби для аутентифікації та авторизації користувачів, кешування, а також має гнучку систему маршрутизації.

Фреймворк пропонує власну ORM для зручної взаємодії з базами даних, а також двигун шаблонів Blade, котрий надає зручні засоби для створення інтерфейсу користувача.

На сьогоднішній день, Laravel - найпопулярніший PHP фреймворк, що продовжує і далі стрімко розвиватися завдяки зусиллям команди розробників. Навколо фреймворку існує потужна екосистема, що пропонує розробнику безліч інших рішень для зручної розробки web-додатків. Це системи моніторингу, бібліотеки, а також хостинг та платформа для розгортання додатків.

На теперішній час більшість web проектів, що розробляються за допомогою фреймворків, створюються саме з використанням Laravel.

3.4 MySQL

В даний час системи керування базами даних (СКБД) є важливим інструментом в багатьох областях. В реляційних СКБД дані зберігаються в окремих таблицях і завдяки цьому досягається виграш в швидкості та гнучкості. Одним з найбільш популярних представників реляційних СКБД є MySQL. Дане сховище є багатопотоковою системою та має архітектуру типу клієнт-сервер. MySQL має велику швидкість, є стійким та легким у використанні [10].

СКБД MySQL складається з SQL-сервера, програмних клієнтів, та засобів адміністрування. MySQL є потужним інструментом що забезпечує широкий спектр можливостей адже він підтримує зовнішні ключі, тригери, збережені процедури, відображення та транзакції.

Дана СКБД є програмним забезпеченням з відкритим кодом. Це означає, що дане ПЗ може використовуватись безкоштовно, тому будь-хто може його застосовувати та модифікувати його код відповідно до своїх потреб. Завдяки тому, що MySQL це Open Source проект, він є дуже популярним.

При використанні MySQL на Unix-серверах, що підтримують багатопотоковість, продуктивність СКБД буде вищою. Завдяки своїй доступності, швидкості і безпеці MySQL є чудовим вибором для використання.

3.5 Javascript

JavaScript є переважно клієнтською мовою, зазвичай використовується для роботи на стороні клієнта. З його допомогою можна розробляти програми з найрізноманітнішим функціоналом. Це може бути: аналог механічного годинника, різна анімація, графічні ефекти і багато іншого.

Сьогодні складно переоцінити роль JavaScript в інтернеті. Згідно зі статистикою w3techs, сьогодні понад 95% сайтів застосовують JS [11]. Неймовірна популярність мови робить його одним з найбажаніших для вивчення та для використання у розробці програмних систем.

ChartJs

ChartJS - Javascript бібліотека, що дозволяє будувати красиві плоскі графіки та діаграми. Білбіотека використовує елемент HTML5 Canvas для візуалізації та підтримує всі сучасні браузери (IE11 +).

Діаграми ChartJS є адаптивними за замовчуванням. Вони добре працюють на мобільних пристроях та планшетах. Бібліотека пропонує 8 різних типів діаграм (лінія, смужка, радіолокатор, пончик і пиріг, полярна зона, міхур, розсіювання, зона), а також можливість їх змішування. Усі діаграми анімовані та настроюються.

Leaflet

Leaflet - одна з найпопулярніших бібліотек JavaScript з відкритим кодом для інтерактивних карт. Підтримує більшість мобільних і стаціонарних платформ з числа тих, що підтримують HTML5 та CSS3.

Поряд з OpenLayers і Google Maps API - одна з найбільш популярних картографічних JavaScript-бібліотек, що використовується на таких великих сайтах, як Flickr, Foursquare, Craigslist, Data.gov, IGN, проектах Вікімедіа, OpenStreetMap, Meetup, WSJ, MapBox, CloudMade, CartoDB та інших. Автор бібліотеки - киянин Володимир Агафонкін.

Leaflet дозволяє розробнику, не знайомому з ГІС, легко відображати растрові карти, що складаються з маленьких фрагментів - тайлів, з, можливо, додатковими шарами, що накладаються поверх основного. Шари можуть бути інтерактивними, наприклад, відображати підказку при натисканні по маркеру.

jQuery

jQuery – це JavaScript бібліотека, призначена для зручної взаємодії JavaScript і HTML. Бібліотека дозволяє легко звертатися до елементів DOM і маніпулювати ними. jQuery дозволяє легко писати потужні програми на JavaScript і створювати привабливі анімаційні ефекти. Також jQuery має функціонал для Ajax та кросбраузерності.

Для спрощення виконання повсякденних завдань розробники взялися створювати бібліотеки JavaScript, які часто називають JavaScript-фреймворками. Існує досить багато бібліотек JavaScript. Однією з найпопулярніших є jQuery.

Дана бібліотека створена щоб спростити використання мови JavaScript і зробити його більш доступним як для новачків, так і для розробників зі стажем за рахунок надання простих у застосуванні функцій, що полегшують вирішення повсякденних завдань.

jQuery дає можливість спростити написання JavaScript коду і не турбуватися про нюанси кросбраузерності.

Переваги бібліотеки:

— Отримання доступу до будь-якого елементу (набору елементів);

— Звернення до атрибутів і вмісту елементів DOM і маніпуляція ними;

— Синтаксис селекотора елементів схожий з CSS;

— Надання зручного API для роботи з AJAX;

— Можливість анімації отриманих елементів на сторінці;

— Надзвичайно простий синтаксис;

— Можливість послідовного виконання методів в ланцюжку;

— Проста архітектура модулів, що розширюють базові можливості бібліотеки;

— Величезна мережеве співтовариство користувачів;

— Чудова документація;

— Корисні розширення, такі як jQuery UI які надають додаткову функціональність.

Ще одною перевагою jQuery є те, що стає простіше писати код на JavaScript, який працює на багатьох різних браузерах. Несумісність між популярними браузерами, таких як IE, Firefox і Safari означає, що вам часто доводиться писати різні шматки коду JavaScript для кожного браузера.

jQuery відмінно підходить для:

— Додавання анімаційних ефектів до елементів. jQuery дозволяє легко додавати ефекти, такі як згасання, а також розширення / стиснення;

— Створення XML (Ajax) запитів. Вони використовують JavaScript, щоб запросити дані з веб-сервера без перезавантаження сторінки;

— Створення слайд-шоу зображень;

— Створення меню, що випадає. jQuery дозволяє легко створювати багаторівневі випадаючі меню з анімацією;

— Створення drag & drop інтерфейсів. Використання jQuery для створення сторінки з елементами, положення яких можна змінити шляхом перетягування;

— Додавання функціональності формам. З jQuery ви легко можете додати функціональності форм, створювати текстові поля з автозаповненням з допомогою Ajax, що дозволить отримувати дані з бази даних на сервері.

Звичайно, можна робити все вище перераховане, написавши власний код на JavaScript, але набагато легше просто використовувати jQuery а не винаходити колесо.

jQuery є безкоштовною для завантаження і використання, поширюється по ліцензіях MIT і GPL. До бібліотеки існують сотні плагінів які можна додати для розширення її функціональності.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Для реалізації клієнтського додатку було обрано клієнтно-серверну архітектуру. Обрана архітектура дозволяє організувати програмний агент таким чином, що він буде незалежним від інших систем, що задовольняє принципам багатоагентної системи.

Перевагою клієнт-серверної архітектури є зниження мережевого трафіку при виконанні запитів до бази даних. Наприклад, при необхідності вибору п'яти записів з таблиці, яка містить мільйон, клієнтський додаток посилає серверу запит, який сервером компілюється і виконується, після чого результат запиту (ті самі п'ять записів, а не вся таблиця) передається назад на клієнт.

Іншою перевагою архітектури клієнт / сервер є можливість зберігання бізнес-правил на сервері, що дозволяє уникнути дублювання коду в різних додатках, що використовують загальну базу даних. Крім того, в цьому випадку будь-яке редагування даних може бути зроблено тільки в рамках цих правил.

На рисунку 4.1 зображена архітектура системи.

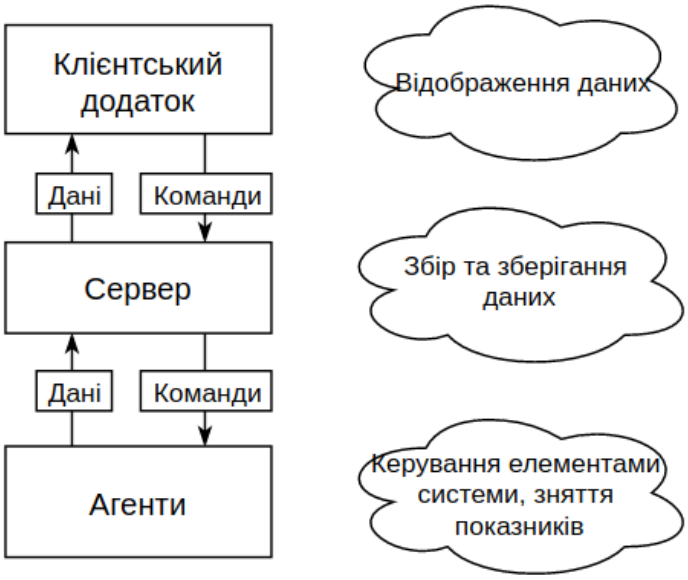


Рисунок 4.1 — Архітектура програмної частини комплексу

Вимоги до клієнтського додатку зображені на Use-case діаграмі (рисунок 4.2). В системі повинно бути два типи акторів - авторизований користувач та неавторизований користувач. Авторизований користувач має більше можливостей, зокрема може редагувати інформацію про агенти а також додавати в систему нових агентів. Для авторизованого користувача доступна сторінка статистики системи.

Користувачі повинні мати можливість переглядати інформацію про стан метеорологічних умов на дорогах а також прогнози для ділянок дороги.

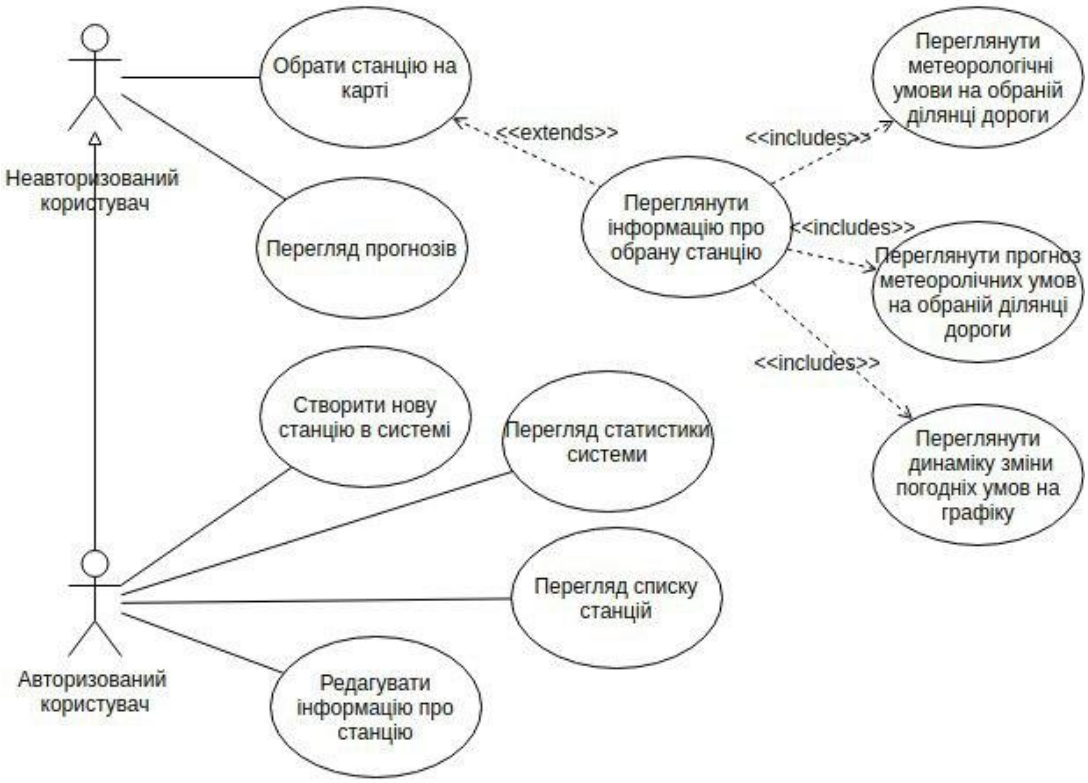


Рисунок 4.2 — Use-case діаграма клієнтського додатку

4.1 MVC

Архітектурна модель Model-View-Controller (MVC) розділяє додаток на три основні групи компонентів: моделі даних, вигляди (інтерфейс користувача) та модулі керування (контролери). За допомогою цього шаблону запити користувачів перенаправляються до Контролера, який відповідає за роботу з Моделями для

виконання дій користувача та / або отримання результатів запитів. Контролер вибирає Вигляд для відображення користувачу та надає йому потрібні дані Моделі.

Застосування даного шаблону допомагає досягти розмежування даних, інтерфейсу користувача та логіки системи. Це розділення призводить до того, що зміни інтерфейсу користувача мінімально впливають на роботу з даними, а зміни в моделі даних можуть виконуватись без змін інтерфейсу користувача.

Дане розмежування обов'язків допомагає масштабувати додаток за складністю, оскільки простіше кодувати, налагоджувати та тестувати щось (модель, перегляд чи контролер), що має єдине завдання. Складніше змінити, протестувати і налагодити код, який має залежності, розподілені по двох або більше з цих трьох областей. Наприклад, логіка користувацького інтерфейсу, як правило, змінюється частіше, ніж логіка бізнесу. Якщо код вигляду та бізнес-логіка поєднуються в одному об'єкті, об’єкт, що містить бізнес-логіку, повинен змінюватися щоразу, коли змінюється інтерфейс користувача. Це часто призводить до виникнення помилок та вимагає повторної перевірки логіки після кожної мінімальної зміни інтерфейсу користувача.

На рисунку 4.3 показані три основні компоненти шаблону MVC.

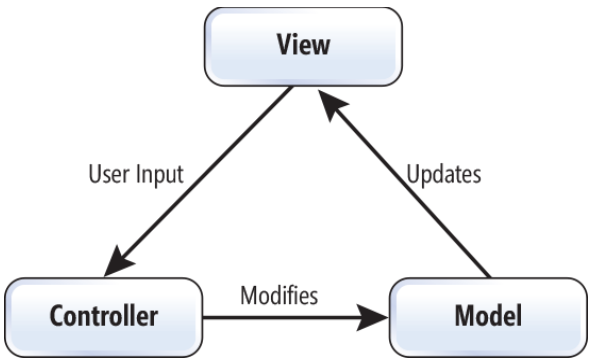


Рисунок 4.3 — Принцип роботи шаблону MVC

4.2 Структура проекту

Структура проекту типова для MVC фреймворку. Моделі, що представляють сутності в базі даних наслідують абстрактний клас Model. Класи, що позначають метеорологічні показники наслідують базовий клас WeatherCharacteristic. Для

позначення сутностей попереджень створена власна ієрархія. Попередження Warning бувають двох видів - про поточний стан на дорозі (PresentWarning) та про майбутній стан дороги, заснований на прогнозах FutureWarning. На рисунку 4.4 зображена спрощена UML-діаграма класів системи.

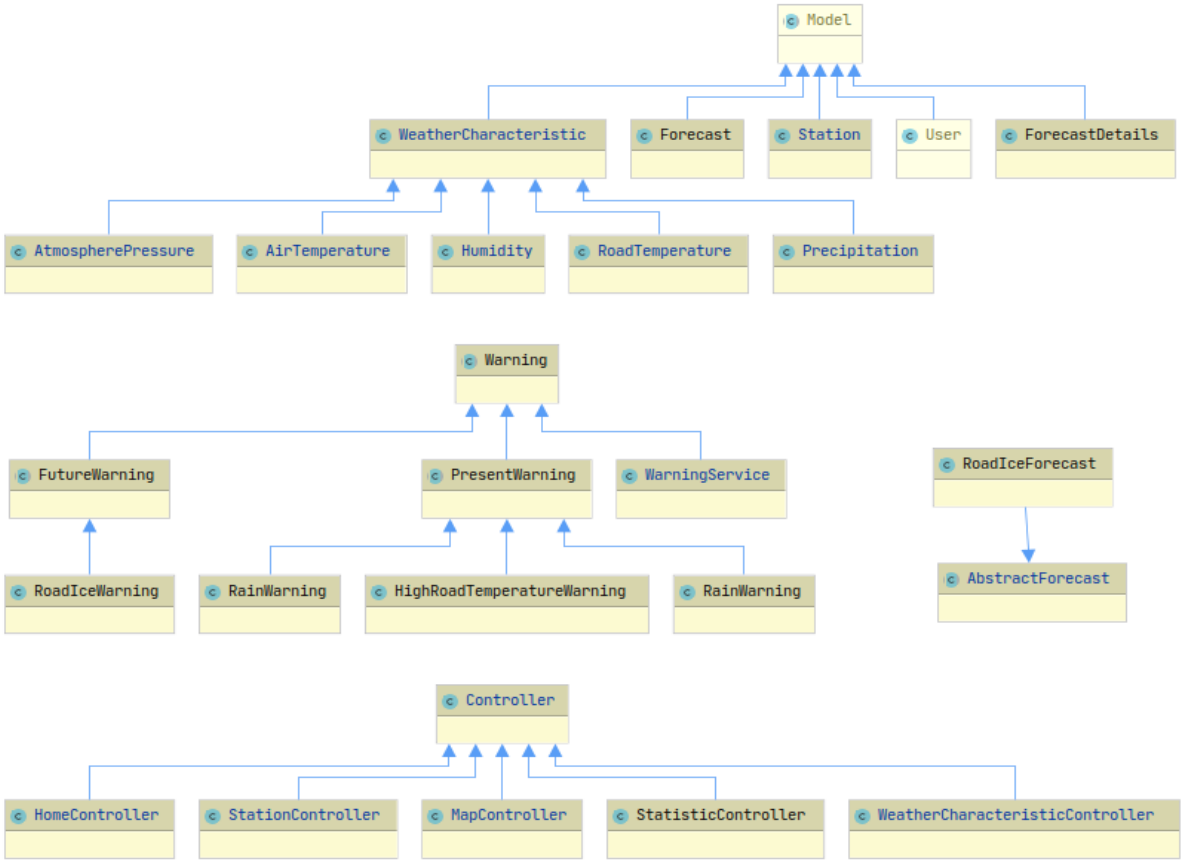


Рисунок 4.4 — UML - діаграма класів

4.3 Опис бази даних

База даних складається з семи таблиць (рисунок 4.5).

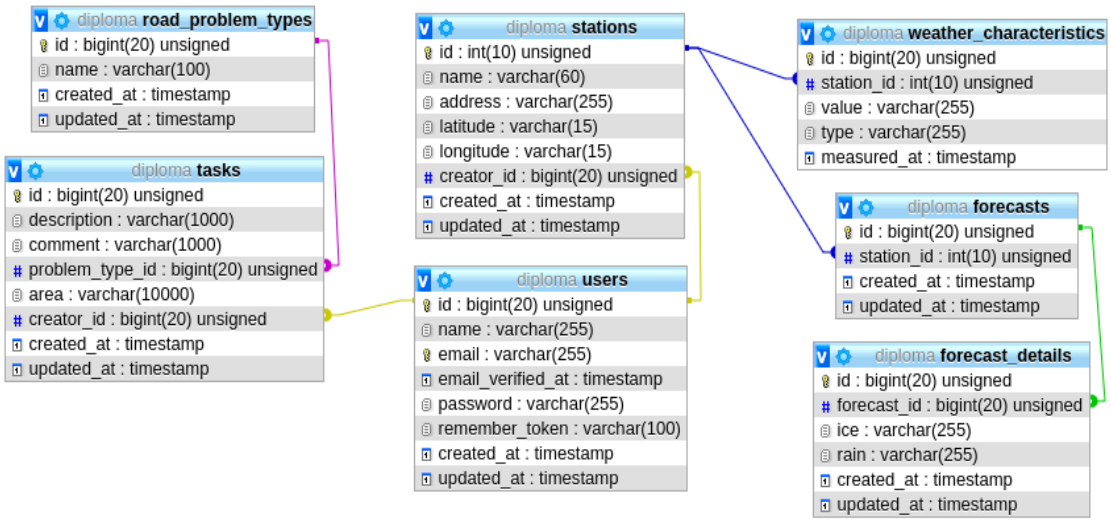


Рисунок 4.5 — Структура бази даних

Для дотримання цілісності бази даних, кожна таблиця має первинний ключ. Проведена нормалізація, за допомогою якої було вирішено проблему надмірності представлення даних, що в свою чергу потенційно призвело б до логічних помилок вибірки при зміні даних. Використовуються зовнішні ключі. Отримана модель задовольняє трьом нормальним формам.

Таблиця «stations» містить інформацію про агенти моніторингу і є основною таблицею. Вона має наступні поля : id, назва, адреса, широта, довгота.

В таблиці «users» зберігаються дані про користувачів: id, ім’я, email та ін. Таблиця пов’язана з таблицею «stations» зовнішнім ключем по полю creator_id.

Таблиця «forecasts» зберігає інформацію про прогнози, зроблені в системі. Вона пов’язана зі «stations» зовнішнім ключем по полю station_id. Детальна інформація про прогноз винесена в окрему таблицю «forecast_details», зв’язок з якою відбувається по полю forecast_id.

В таблиці «weather_characteristics» зберігаються дані про метеорологічні показники: id, id агента, значення показника та його тип. Вона пов’язана зі «stations» зовнішнім ключем по полю station_id.

Таблиці «road_broblem_types» та «tasks» призначені для зберігання сервісної інформації.

4.4 Опис окремих функцій системи

Попередження

Система здатна повідомляти про поточний стан умов на дорогах. Повідомлення виводиться на сторінці агенту. Висновок про поточний стан умов робиться на основі даних, отриманих з агентів моніторингу метеорологічних показників. Також для цього можуть використовуватись сторонні системи, наприклад, інформація про деякі погодні характеристики може бути отримана з системи Гідрометцентру.

Дана інформація є корисною для водіїв. У разі небезпечних умов на дорозі, водій після того як побачить повідомлення на сторінці системи, може вжити необхідні заходи для забезпечення своєї безпеки на дорозі.

Окрім повідомлень про поточний стан дорожнього покриття на сторінці агенту також наявні повідомлення- попередження про несприятливі погодні умови, що наближаються.

Приклади повідомлень:

— Висококa температура дорожнього покриття;

— Мокре дорожнє полотно(дощ) ;

— Ожеледиця;

— Погана видимість(туман, задимлення).

На основі вище перерахованих умов система робить загальний висновок про безпеку на дорозі.

Система розроблена таким чином, що є можливість додавати логіку нових повідомлень, створивши новий клас і вказавши в ньому необхідні умови для даного повідомлення. Для додавання нової несприятливої умови (наприклад погана видимість на дорозі) необхідно указати правила згідно з якими система буде вважати, що дана несприятлива умова наявна(наприклад показник видимості, отриманий з відповідного давача менше 50%).

Прогнозування

Для певних ділянок дороги є актуальною задача прогнозу погодних умов. Прогноз в системі виконується при запуску консольної команди, котра запускається за розкладом.

Логіка прогнозування знаходится у класах похідних від Forecast. Додавати нові прогнози можна, розширивши клас Forecast. Для здійснення прогнозування у класі прогнозу повинна бути задана логіка виконання прогнозу. Для цього може бути виконана оцінка динаміки зміни метеопоказників. Так, наприклад, для прогнозування ожеледиці система аналізує зміну температури, вологості та атмосферного тиску. Якщо дана динаміка відповідає умовам виникнення несприятливої умови, тоді відбувається розрахунок її ймовірності виникнення.

Прогнози виводяться на сторінці(рис. 4.6), на якій відображено список агентів та прогнози щодо метеоумов на відповідних дорожніх ділянках

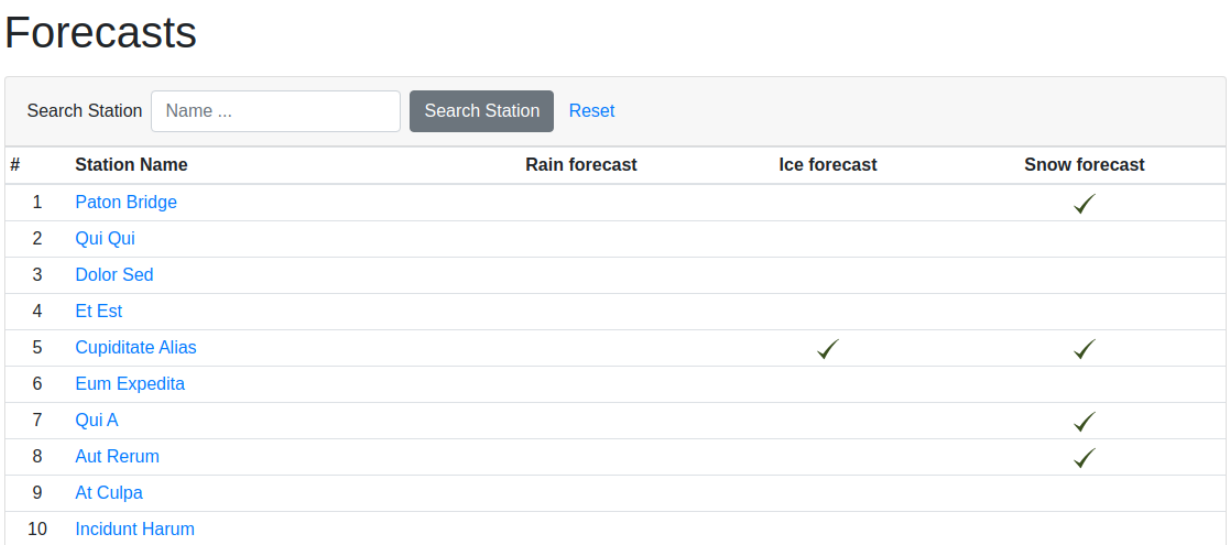


Рисунок 4.6 — Вікно перегляду прогнозів

Реагування

У випадку коли система розуміє що згенеровано певне попередження або прогнозування виявило появу несприятливих погодних умов, система може зреагувати та виконати указані дії, що указані у коді програми.

Реагуючи на дані події, система може звертатися до сторонніх сервісів та викликати їх команди.

Приклад: коли система спрогнозувала, що скоро буде ожеледиця, система звернеться до сервісу дорожньої служби і виконає запит на виклик спеціалізованого автомобіля що посипає дороги солевою сумішшю на прилеглій ділянці.

Ще один приклад: система виявила високу температуру дорожнього покриття. Вона реагує на це – повідомляє необхідні служби про подію. Служби приймають рішення про зменшення максимальної допустимої ваги автомобілів, що проїжджають на даній дорозі.

Розроблена система здатна взаємодіяти з системами попередження. У разі генерації попереджень про несприятливі умови, додаток сигналізує системи дорожнього попередження і ті діють певним чином. Наприклад, можуть виводити інформацію на табло, попереджаючи водіїв про небезпеку.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

На головній сторінці системи знаходиться карта з мітками, що позначають агентів моніторингу. При натисканні на мітку спливає вікно (рисунок 5.1) з короткою інформацією про даний агент: назва, координати та стан погодних умов на даній ділянці дороги. Для переходу до вікна, що містить інформацію про даний агент, необхідно натиснути на назву агента. Якщо користувач авторизований, він має можливість додавати агенти в систему, натиснувши на місце на карті.

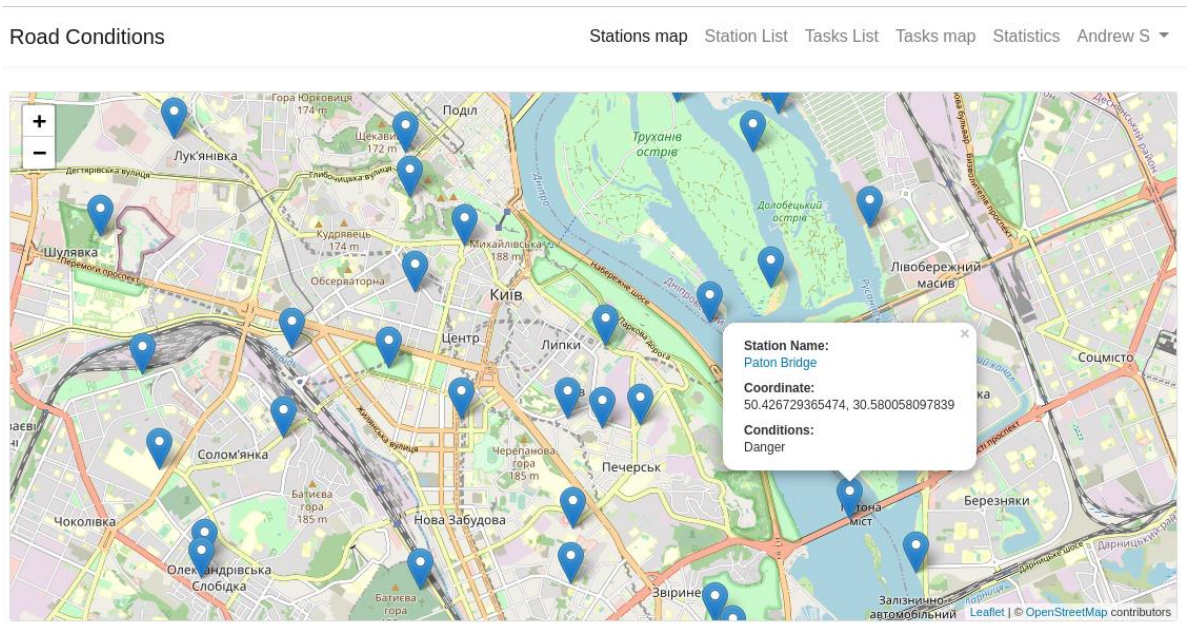


Рисунок 5.1 — Перегляд короткої інформації про агент на карті

На сторінці агента користувач бачить інформацію про нього - назву, координати його місцезнаходження. На карті справа показано розташування агента. Нижче на сторінці знаходиться інформація про стан безпеки метеорологічних умов на ділянці дороги, що вимірюється даним агентом. Якщо умови несприятливі, то також виводяться попередження для водіїв у блоці червоного кольору (рисунок 5.2).

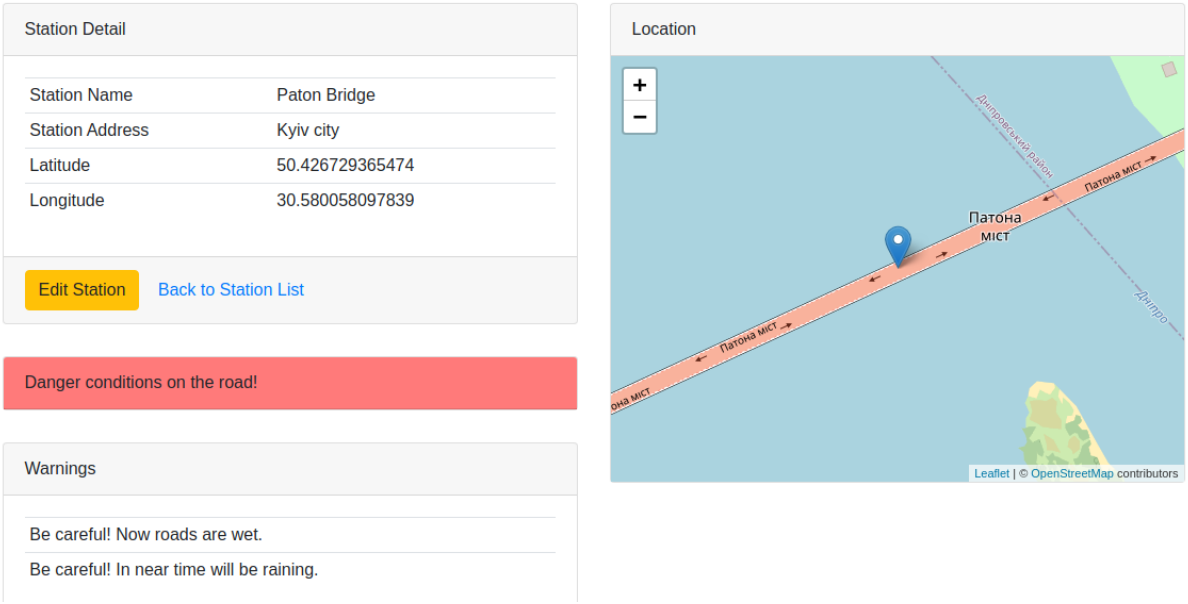


Рисунок 5.2 — Повідомлення про небезпечні умови на сторінці агента

Коли метеорологічні умови на ділянці дороги безпечні для водіння, про це сигналізує повідомлення зеленого кольору (рисунок 5.3).

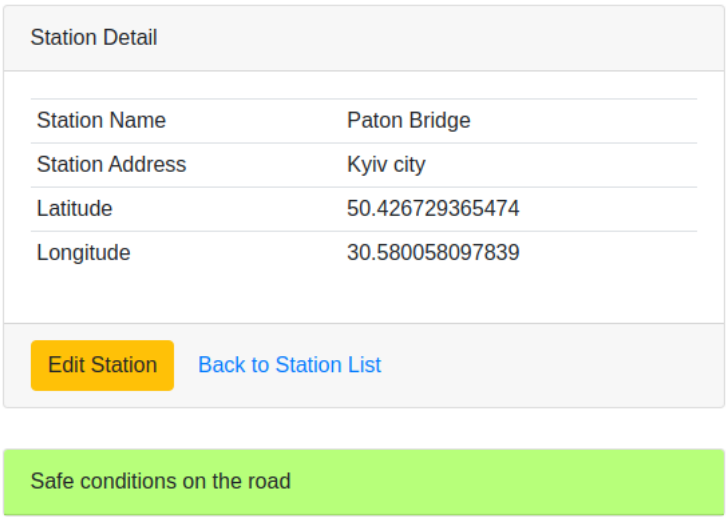


Рисунок 5.3 — Повідомлення про безпечність умов на дорогах

Авторизований користувач може змінювати параметри агента.

Нижче на сторінці розташований графік, що демонструє зміну метеорологічних показників (рисунок 5.4). Діаграма є інтерактивною, що надає користувачу

можливість приховувати на ній графіки показників, натиснувши на позначення зверху діаграми. При наведенні курсора на графік, з'являється вікно в якому відображаються всі показники на обраний момент часу.

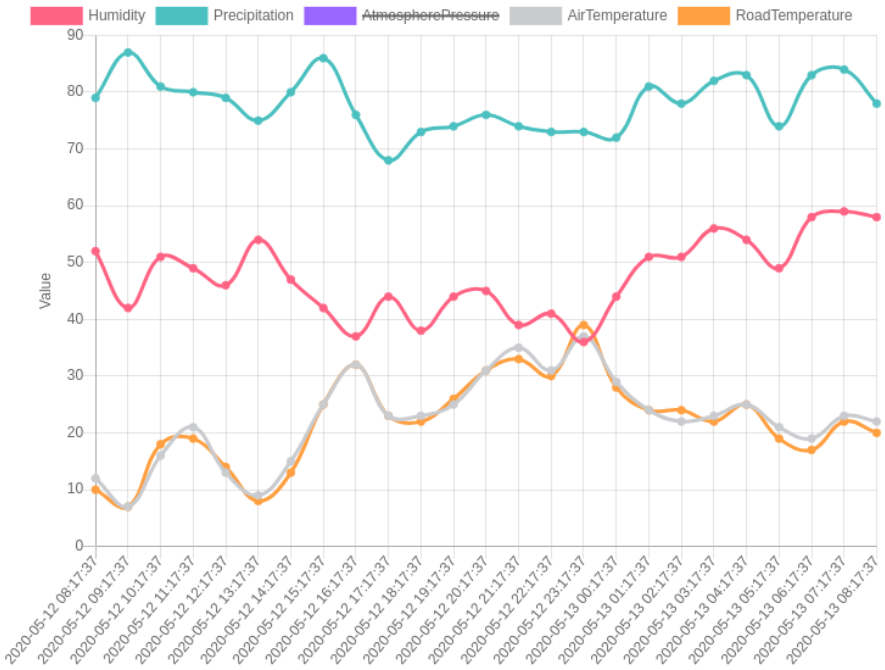


Рисунок 5.4 — Графік зміни метеорологічних показників

Авторизований користувач може переглядати список агентів, виконувати їх пошук по назві. В даному меню відображається наступна інформація: адреса, координати місцезнаходження, а також висновок про метеорологічні умови на дорожньому покритті (вкладка Conditions) (рисунок 5.5). З даного списку, користувач може потрапити на сторінку агента, обравши його в списку.

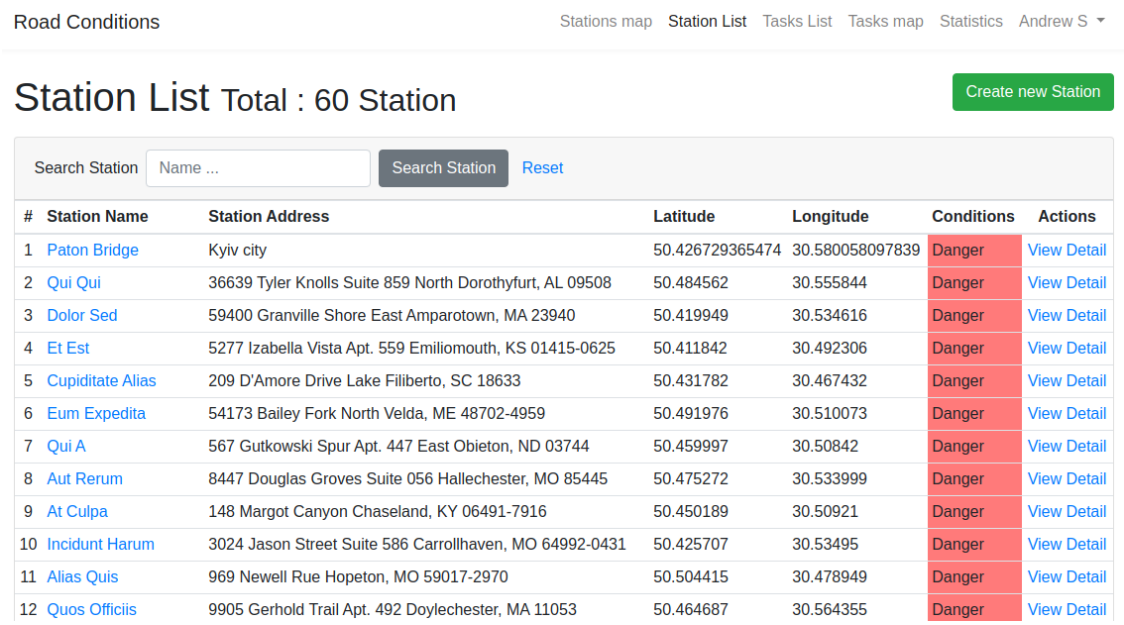
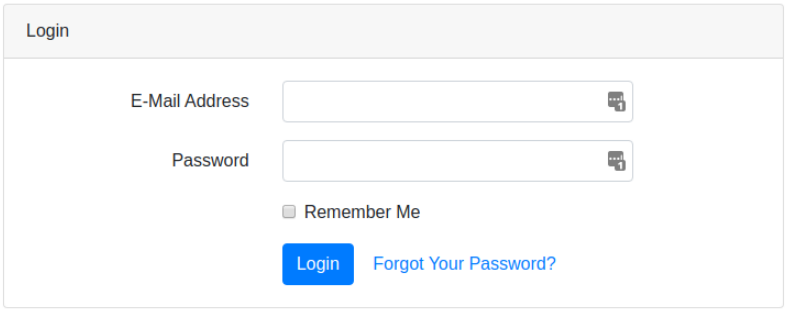


Рисунок 5.5 — Список агентів моніторингу

На сторінці прогнозів відображено список агентів та прогнози щодо метеоумов на відповідних дорожніх ділянках (рисунок 5.6).

На рисунку 5.6 зображено вікно авторизації в системі.



.

Рисунок 5.6 — Форма авторизації в системі

В авторизованого користувача є можливість переглядати статистику в системі (рисунок 5.7).

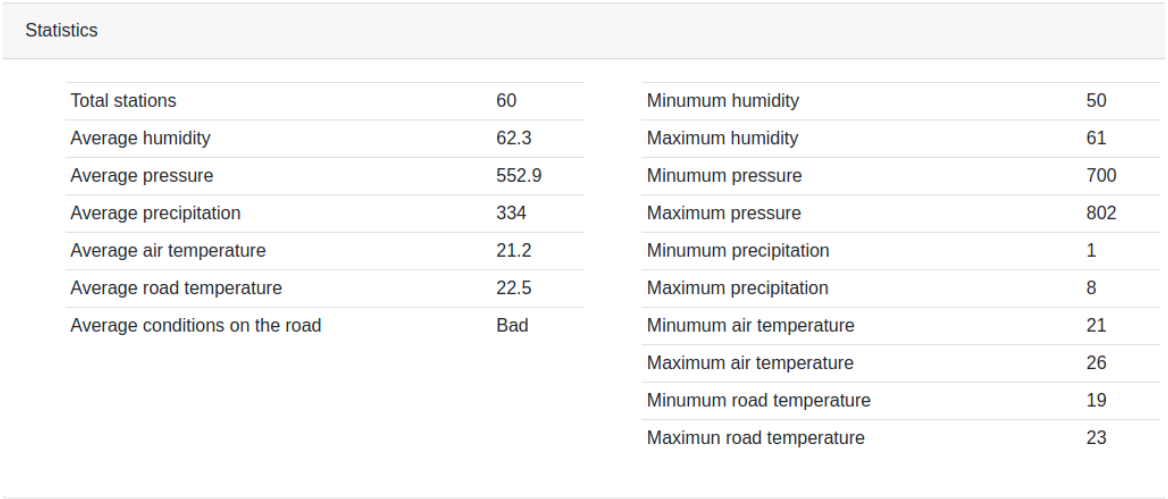


Рисунок 5.7 — Вікно статистики системи

В даному вікні відображаються статистичні дані, а саме - середні, мінімальні та максимальні значення метеорологічних показників. А також кількість агентів в системі та загальний висновок про стан дорожніх умов.

ВИСНОВКИ

Було освоєно принципи побудови багатоагентних систем та використання агентів моніторингу для спостереження за метеорологічними показниками на дорогах.

Проведено огляд засобів розробки апаратно-програмного комплексу.

Розроблений апаратно-програмний комплекс дозволяє спостерігати за метеоумовами на дорожньому покритті та реагувати у разі виникнення несприятливих. Система здатна робити прогнози погодних умов для конкретних агентів.

Під час виконання дипломної роботи було покращено навички розробки клієнтських web-застосунків. Було здобуто вміння програмування мікроконтролерів. Були отримані знання та навички побудови агентів, що входять до складу багатоагентної системи. Також були отримані навички застосування шаблону MVC та архітектури клієнт-сервер.

Розроблений клієнтський додаток дозволяє переглядати виміряні показники, а також прогнози, зроблені системою.

Таким чином чином виконання дипломної роботи поглибило знання різноманітних технологій, зокрема тих, що використовуются для розробки багатоагентних систем, використання яких є невід’ємною частиною сучасних програмних продуктів, а також клієнтських застосунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Антоненко В. М., Рогушина Ю.В. Сучасні інформаційні системи і технології. Навчальний посібник. - К.: КСУ МГІ, 2005. – 131 с.

2. Городецкий В.И., Грушинский М.С., Хабалов А.В. Многоагентные системы (обзор) [Электронный ресурс] / В.И. Городецкий, М.С. Грушинский, А.В. Хабалов. – Режим доступа: http://www.raai.org/library/ainews/1998/2/GGKHMAS.ZIP

3. Боєв, В. Д. Імітаційне моделювання систем : навч. посібник для прикладного бакалавріату / В. Д. Боєв. — М. : Видавництво Юрайт, 2017. — 253 с. — (Серія : Бакалавр. Прикладний курс). ISBN 978-5-534-04734-9

4. Уллі Соммер. Програмування мікроконтролерних плат Arduino / Freeduino.

5. Виктор Петин: Arduino и Raspberry Pi в проектах Internet of Things

6. Адель Джавед. Побудова проектів Arduino для Internet of Things: експерименти з реальними програмами.

7. Джеремі Блум. Вивчаємо Arduino. Інструменти і методи технічного чарівництва.

8. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство. — СПб.: Питер, 2013. — 512 с.: ил.

9. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е изд. — СПб.: Питер, 2016. — 768 с.: ил. — (Серия «Бестселлеры O’Reilly»). ISBN 978-5-496-02146-3

10. Фиайли К. SQL: Пер. с англ. – М.: ДМК Пресс. – 456 с.: ил. (Серия «Quick Start»). ISBN 5-94074-233-5

11. Закас Н. JavaScript для профессиональных веб-разработчиков / [Пер. с англ. А. Люти ча]. -СПб.: Питер, 2015. -960 с.: ил. -(Серия «Для профессионалов»).

12. JavaScript. Подробное руководство. – Пер. с англ. – СПб: СимволПлюс, 2008. – 992 с., ил. ISBN10: 5932861037 ISBN13: 978593286105

13. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям – М.: УРСС, 2002

14. Shoham Y., Leyton-Brown K., Multiagent Systems: Algorithmic, Gamer-Theoretic, and Logical Foundations. Cambridge UP, 2009. http://www.masfoundations.org/

15. Vidal, J.M.: Fundamentals of multiagent systems with NetLogo examples. MIT Press (2010) http://jmvidal.cse.sc.edu/papers/mas.pdf

16. Weiss G., editor. Multi-Agent Systems. MIT Press, 2013, 2nd edition. http://www.the-mas-book.info

17. Расников В.П. Оценка состояния проезжей части дороги в зимний период. - Автомоб. дороги, 1975, № 9.

18. Анохин Б.Б., Аржанухина С.П., Кочетков А.В. Специализированный прогноз состояния дорожного покрытия

ДОДАТОК 1

Апаратно-програмний комплекс моніторингу та управління станом дорожнього покриття Smart City

СПЕЦИФІКАЦІЯ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б

Аркушів 2

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б 12-1		Модулі програмного коду системи
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б 13-1		Опис програмного модулю

ДОДАТОК 2

Апаратно-програмний комплекс моніторингу та управління станом дорожнього покриття Smart City

ТЕКСТ ПРОГРАМНОГО МОДУЛЮ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б-12-1

Аркушів 9

Київ 2020

Текст програмного коду прошивки мікроконтролера

```
#include <EtherCard.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BME280.h>

#define REQUEST_RATE 5000 // milliseconds

Adafruit_BME280 bme; // use I2C interface
Adafruit_Sensor *bme_temp = bme.getTemperatureSensor();
Adafruit_Sensor *bme_pressure = bme.getPressureSensor();
Adafruit_Sensor *bme_humidity = bme.getHumiditySensor();

char *removeSpaces(char *source)
{
   char *target;
   while (*source++ && *target)
   {
      if (!isspace(*source))
         *target++ = *source;
   }
   return target;
}

int rainPin = A0;

// ethernet interface mac address, must be unique on the LAN

static byte mymac[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};
byte Ethernet::buffer[700];
static uint32_t timer;

const char website[] PROGMEM = "website.com";

// called when the client request is complete
static void my_callback(byte status, word off, word len)
{
    Serial.println(">>>");
   Ethernet::buffer[off + 300] = 0;
   Serial.print((const char *)Ethernet::buffer + off);
   Serial.println("...");
}

void setup()
{
   pinMode(rainPin, INPUT);

   if (!bme.begin(0x76))
   {
      Serial.println(F("Could not find a valid BME280 sensor, check wiring!"));
      while (1)
         delay(10);
   }
   bme_temp->printSensorDetails();
   bme_pressure->printSensorDetails();
   bme_humidity->printSensorDetails();
   Serial.begin(9600);

   Serial.println(F("\n[webClient]"));

   // Change 'SS' to your Slave Select pin, if you arn't using the default pin
   if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
      Serial.println(F("Failed to access Ethernet controller"));


   if (!ether.dhcpSetup())
```

```
            Serial.println(F("DHCP failed"));

        ether.printIp("IP:  ", ether.myip);
        ether.printIp("GW:  ", ether.gwip);
        ether.printIp("DNS: ", ether.dnsip);

    #if 1
      // use DNS to resolve the website's IP address
      if (!ether.dnsLookup(website))
        Serial.println("DNS failed");

         memcpy(ether.hisip, websiteip, sizeof(websiteip));

    #elif 0
        // if website is a string containing an IP address instead of a domain name,
        // then use it directly. Note: the string can not be in PROGMEM.
        char websiteIP[] = "192.168.1.1";
        ether.parseIp(ether.hisip, websiteIP);
    #else
        // or provide a numeric IP address instead of a string
        byte hisip[] = {192, 168, 1, 1};
        ether.copyIp(ether.hisip, hisip);
    #endif

        ether.printIp("SRV: ", ether.hisip);
    }

    void loop()
    {
        ether.packetLoop(ether.packetReceive());

        if (millis() > timer)
        {

            int waterSensorValue = analogRead(rainPin);
            sensors_event_t temp_event, pressure_event, humidity_event;
            bme_temp->getEvent(&temp_event);
            bme_pressure->getEvent(&pressure_event);
            bme_humidity->getEvent(&humidity_event);

            Serial.print(F("Temperature = "));
            Serial.print(temp_event.temperature);
            Serial.println(" *C");

            Serial.print(F("Humidity = "));
            Serial.print(humidity_event.relative_humidity);
            Serial.println(" %");

            Serial.print(F("Pressure = "));
            Serial.print(pressure_event.pressure);
            Serial.print(" hPa /");
            Serial.print(pressure_event.pressure / 1000 * 750.1);
            Serial.println(" мм рт ст");
            Serial.print(F("Precipitation = "));
            Serial.println(waterSensorValue);

            char buffer1[400];

            static float f_val = humidity_event.relative_humidity;
            static float f_val2 = pressure_event.pressure / 1000 * 750.1;
            static float f_val3 = temp_event.temperature;
            static float f_val4 = 1020 - waterSensorValue;

            timer = millis() + 5000;
            Serial.println();
            Serial.print("<<< REQ ");
            String s1 = "";
            String s2 = s1 + "?humidity=" + f_val + "&pressure=" + f_val2 + "&air_temperature=" + f_val3 + "&precipitation=" + f_val4
+ "&station_id=2";
```

```
        Serial.println(s2);
        char copy[50];
        s2.toCharArray(copy, 50);
        ether.browseUrl(PSTR("/add_measure"), copy, website, my_callback);
        ether.persistTcpConnection(true);
    }
}
```

Текст програми клієнтського додатку

Лістинг файлу StationController.php — реалізація відображення та управління агентами моніторингу

```
<?php

namespace App\Http\Controllers;

use App\Services\Warnings\WarningService;
use App\Station;
use App\WeatherCharacteristic;
use Illuminate\Http\Request;

class StationController extends Controller
{
    /**
     * Display a listing of the station.
     *
     * @return \Illuminate\View\View
     */

    public function index()
    {
//        $this->authorize('manage_station');

        $stationQuery = Station::query();
        $stationQuery->where('name', 'like', '%'.request('q').'%');
        $stations = $stationQuery->paginate(25);
        foreach ( $stations as &$station) {
            $warningService = new WarningService($station->id);
            $warnings = $warningService->getWarnings();
            $status = 0;//all is ok
            if ($warnings) $status = 1; // danger
            $statusColors = [
                '#75ff0085',
                '#ff000085'
            ];
            $statusTexts = [
                'Safe conditions on the road',
                'Danger conditions on the road!',
            ];
            $station->status = $status ? 'Danger' : 'Safe';
            $station->statusColor = $statusColors[$status];
        }

        return view('stations.index', compact('stations'));
    }

    /**
     * Show the form for creating a new station.
     *
     * @return \Illuminate\View\View
     */
    public function create()
    {
        $this->authorize('create', new Station);

        return view('stations.create');
```

```
    }

    /**
     * Store a newly created station in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Routing\Redirector
     */
    public function store(Request $request)
    {
        $this->authorize('create', new Station);

        $newStation = $request->validate([
            'name'      => 'required|max:60',
            'address'   => 'nullable|max:255',
            'latitude'  => 'nullable|required_with:longitude|max:15',
            'longitude' => 'nullable|required_with:latitude|max:15',
        ]);
        $newStation['creator_id'] = auth()->id();

        $station = Station::create($newStation);

        return redirect()->route('stations.show', $station);
    }

    /**
     * Display the specified station.
     *
     * @param  \App\Station  $station
     * @return \Illuminate\View\View
     */
    public function show(Station $station)
    {
        $res = WeatherCharacteristic::where('station_id', $station->id)->get();

        $weatherData = $res->groupBy('type');
        $dataForCharts = [];

        $warningService = new WarningService($station->id);
        $warnings= $warningService->getWarnings();
        $status = 0;//all is ok
        if($warnings)$status = 1; // danger
        $statusColors =[
            '#75ff0085',
            '#ff000085'
        ];
        $statusTexts =[
            'Safe conditions on the road',
            'Danger conditions on the road!',
        ];
        return view('stations.show', ['station'=>$station,'measureData'=>$weatherData,'warnings'=>$warnings,'statusColor'=>$statusColors[$status],'statusText'=>$statusTexts[$status]]);
    }

    /**
     * Show the form for editing the specified station.
     *
     * @param  \App\Station  $station
     * @return \Illuminate\View\View
     */
    public function edit(Station $station)
    {
        $this->authorize('update', $station);

        return view('stations.edit', compact('station'));
    }

    /**
```

```
     * Update the specified station in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Station  $station
     * @return \Illuminate\Routing\Redirector
     */
    public function update(Request $request, Station $station)
    {
        $this->authorize('update', $station);

        $stationData = $request->validate([
            'name'      => 'required|max:60',
            'address'   => 'nullable|max:255',
            'latitude'  => 'nullable|required_with:longitude|max:15',
            'longitude' => 'nullable|required_with:latitude|max:15',
        ]);
        $station->update($stationData);

        return redirect()->route('stations.show', $station);
    }

    /**
     * Remove the specified station from storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Station  $station
     * @return \Illuminate\Routing\Redirector
     */
    public function destroy(Request $request, Station $station)
    {
        $this->authorize('delete', $station);

        $request->validate(['station_id' => 'required']);

        if ($request->get('station_id') == $station->id && $station->delete()) {
            return redirect()->route('stations.index');
        }

        return back();
    }
}
```

Лістинг файлу Forecast.php — реалізація окремого пронозу

```
<?php

namespace App\Services\Forecasts;

use App\Forecast;
use App\ForecastDetails;
use App\Station;

class ForecastService
{
    private $forecasts;

    public function __construct()
    {
        $this->forecasts = [
            RoadIceForecast::class
        ];
    }

    public function run()
    {
        ForecastDetails::query()->delete();
        Forecast::query()->delete();

        $stations = Station::all();
```

```
        foreach ($stations as $station) {
            $forecast = Forecast::create(['station_id' => $station->id]);
            $forecastModel = new  ForecastDetails(['forecast_id' => $forecast->id]);
            foreach ($this->forecasts as $forecastClass) {
                (new  $forecastClass($forecastModel, $station->id))->run();
            }
            $forecastModel->save();
        }
    }
}
```

Лістинг файлу ForecastService.php — клас відповідає за запуск прогнозування

```
<?php

namespace App\Services\Forecasts;

use App\Forecast;
use App\ForecastDetails;
use App\Station;

class ForecastService
{
    private $forecasts;

    public function __construct()
    {
        $this->forecasts = [
            RoadIceForecast::class
        ];
    }

    public function run()
    {
        ForecastDetails::query()->delete();
        Forecast::query()->delete();

        $stations = Station::all();
        foreach ($stations as $station) {
            $forecast = Forecast::create(['station_id' => $station->id]);
            $forecastModel = new  ForecastDetails(['forecast_id' => $forecast->id]);
            foreach ($this->forecasts as $forecastClass) {
                (new  $forecastClass($forecastModel, $station->id))->run();
            }
            $forecastModel->save();
        }
    }
}
```

Лістинг файлу Warning.php — реалізація попередження

```
<?php

namespace App\Services\Warnings;

abstract class  Warning
{
    public $stationId;

    public function __construct($stationId)
    {
        $this->stationId = $stationId;

    }

    abstract public function condition();

    abstract public function get();
```

```
}
```

Лістинг файлу WarningService.php —клас відповідає за логіку створення та управління попередженнями

```
<?php

namespace App\Services\Warnings;

use App\Services\Warnings\Future\RoadIceWarning;
use App\Services\Warnings\Present\HighRoadTemperatureWarning;

class WarningService
{

   public $warningsAboutFuture;
   public $warningsAboutPresent;
   public $stationId;

   public function __construct($stationId)
   {
      $this->stationId = $stationId;
      $this->warningsAboutFuture = [
         'App\Services\Warnings\Future\RainWarning',
         RoadIceWarning::class
      ];

      $this->warningsAboutPresent = [
         HighRoadTemperatureWarning::class,
         'App\Services\Warnings\Present\RainWarning',
      ];
   }

   public function getWarnings()
   {
      $presentWarnings = [];
      foreach ($this->warningsAboutPresent as $warningClass) {
         $warning = (new $warningClass($this->stationId))->get();
         if ($warning)
            $presentWarnings[] = $warning;
      }

      $futureWarnings = [];
      foreach ($this->warningsAboutFuture as $warningClass) {
         $warning = (new $warningClass($this->stationId))->get();
         if ($warning)
            $presentWarnings[] = $warning;
      }

      return array_merge($presentWarnings, $futureWarnings);
   }
}
```

Лістинг файлу WeatherCharacteristicController.php —клас призначений для отримання даних з мікроконтролера та їх запису в базу даних

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\WeatherCharacteristic;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class WeatherCharacteristicController extends Controller
{
   /**
    * Get outlet listing on Leaflet JS geoJSON data structure.
    *
    * @param \Illuminate\Http\Request $request
```

```
     * @return bool
     */
    public function store(Request $request)
    {
        Log::info($_REQUEST);
        $dataRequest =[
            ['value'=>$request->get('humidity'),'type'=>'\App\Humidity'],
            ['value'=>$request->get('pressure'),'type'=>'\App\AtmospherePressure'],
            ['value'=>(int)($request->get('precipitation')/10),'type'=>'\App\Precipitation'],
            ['value'=>$request->get('airTemperature'),'type'=>'\App\AirTemperature'],
            ['value'=>$request->get('roadTemperature',$request->get('airTemperature')+1),'type'=>'\App\RoadTemperature']
        ];

        $station_id = $request->get('station_id');
        $measured_at = $request->get('measured_at',now());

        $measures  =$request->get('measures',$dataRequest);
        $data = [];
        foreach ($measures as $measure){
            $data[] =[
                'station_id' =>$station_id,
                'value' => $measure['value'],
                'type' => $measure['type'],
                'measured_at' => $measured_at
            ];
        }
        WeatherCharacteristic::insert($data);

        return true;
    }
}
```

ДОДАТОК 3

Апаратно-програмний комплекс моніторингу та управління станом дорожнього покриття Smart City

ОПИС ПРОГРАМНОГО МОДУЛЮ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ТВ_6148_20Б-13-1

Аркушів 6

Київ 2020

АНОТАЦІЯ

Розроблений програмний продукт забезпечує моніторинг та управління стану дорожнього покриття.

В дипломній роботі було розроблено програмний агент моніторингу на базі мікроконтролера Arduino Mega та агент управління з клієнтським додатком. Розробка була виконана з використанням засобів C, Arduino IDE, MySQL, PHP, Javascript.

ЗМІСТ

1. ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Програмний код прошивки мікроконтролера, на базі якого побудований агент моніторингу, розроблено у середовищі Arduino IDE, використовуючи мову програмування С.

Основне завдання прошивки - знімати інформацію про метеорологічні показники з давачів, що під'єднані до мікроконтролера та відправляти цю інформацію на агент управління за допомогою GSM підключення до мережі Internet.

При створенні програмного продукту були використані:

— середовище розробки PhpStorm, що надає потужні інструменти для розробки програмного забезпечення web застосунків;

— мова програмування PHP;

— фреймворк Laravel , що організує проект у зручну структуру а також надає необхідні та корисні засоби для розробки;

— мова розмітки HTML;

— каскадні таблиці стилів CSS для кастомізації користувацького інтерфейсу;

— мова програмування Javascript для програмування інтерактивного та дружнього до користувача інтерфейсу;

— систему керування базами даних MySQL для зберігання даних, якими оперує система;

— Javascript бібліотека jQuery, що призначена для спрощеної маніпуляції DOM деревом;

— Javascript бібліотека Leaflet для використання інтерактивних карт з потужними можливостями кастомізації;

— Javascript бібліотека Chart.js для побудови візуалізацій даних у вигляді графіків та діаграм;

— система контролю версії GIT;

— шаблонізатор Blade, що забезпечує зручне розділення шару даних та шару представлення застосунку.

1.1 Опис логічної структури

Прорамний продукт складаєтсья з двох частин: коду прошивки мікроконтролеру та коду клієнтського додатку. Код прошивки використовує бібліоткеки для роботи з датчиками а також бібліотеку для модулю зв’язку, котрий дозволяє агенту комунікувати з агентом контролю.

Для реалізації клієнтського додатку були використані мова PHP, база даних MySQL, середовище розробки PhpStorm, мова Javascript, каскадні стилі CSS та мова розмітки HTML.

Для реалізації задачі моніторингу та управління метеорологічних умов на дорожньому покритті було вирішено розробляти компоненти як частини багатоагентної системи. Клієнтський додаток розроблено з використанням клієнт-серверної архітектури. Таким чином сервер може бути розгорнутий у будь-якому середовищі, а у якості клієнту виступає браузер користувача.

Інтерфейс користувача розроблено з використанням технологій HTML - мови розмітки, каскадних таблиць стилів CSS, мови Javascript, що дозволяє зробити додаток інтерактивним.

1.2. Вхідні та вихідні дані

Виідними даними для контролеру є дані отримані з датчиків.

Вхідними даними для серверного додатку є команди з клієнтского додатку та дані з контролеру.

Вихідними даними серверного дадатку є результати прогнозів, згенеровані попереджнення, певні реакції системи та інші дані.

Вхідними даними для клієнтського додатку є команди, отримані від котристувача та дані з серверного додатку.

Вихідними даними клієнтського додатку є результати дій системи у відповідь на команди користувача.

2. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано в браузері Google Chrome 83.0.4103.97 на персональному комп’ютері, який працює на базі процесору x64 Intel Core i3 (6th Gen) та має 8 Гб оперативної пам’яті. Розроблене програмне забезпечення є кросплатформенним та кросбраузереним, що дозволяє запускати його на різних операційних системах комп’ютерів будь-якої потужності та в будь-яких сучасних браузерах.