МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **“ Інформаційна система моніторингу онлайн-платформ поточного аналізу метео-даних ”**

Виконав: ст. гр. Іт-62_____

Спеціальності 126 – «Інформаційні системи та технології»_____
(шифр і назва)

_____Гутиряк Антон Володимирович_____
(Прізвище та ініціали)

Керівник: __к.т.н., доц. Луб П.М._____
(Прізвище та ініціали)

Рецензенти: _____
(Прізвище та ініціали)

_____
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

другий (магістерський) рівень вищої освіти
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Гутиряку Антону Володимировичу

1. Тема роботи: «Інформаційна система моніторингу онлайн-платформ поточного аналізу метео-даних»

Керівник роботи Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету 28.04.2023 року № 133/к-с.
2. Строк подання студентом роботи 15.01.2024 р.

3. Початкові дані до роботи: 1. Науково-технічна і довідкова література. 2. Стандарти веб-розробки та структура фреймворків. 3. Методологія побудови Entity-relationship model, Entity-relationship diagram, Static Structure diagram.

4. Зміст розрахунково-пояснювальної записки:

1. *Аналіз передумов застосування IT для управління ризиком у проектах*
2. *Проектування взаємодії клієнта із сервером онлайн-сервісу погоди за допомогою API*
3. *Проектування веб-додатку відображення зведених даних онлайн-сервісів стеження та прогнозу погоди*
4. *Практичне використання інформаційної системи онлайн-платформи аналізу метеоданих*
5. *Охорона праці та безпека в надзвичайних ситуаціях.*

Висновки та пропозиції.
Бібліографічний список.
Додатки.

5. Перелік графічного матеріалу: 1 та 2 – Тема, мета, завдання роботи; 3 – Аналіз популярних сайтів прогнозу погоди; 4 – Аналіз глобальних метеосервісів прогнозу погоди із підтримкою API; 5 – Алгоритм взаємодії клієнтської частини сервісу із серверною; 6 – Сценарій використання API із глобальних метеосервісів; 7 – Проектування інформаційної системи відображення погоди з різних метеосервісів; 8 – Діаграми програмної реалізації інформаційної системи; 9 – Фрагменти програмної реалізації; 10 – Результати практичного використання ІС; 11 – Порівняльна оцінка достовірності онлайн-платформ поточного аналізу метеоданих; 12 – Висновки.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання 28.04.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	*28.04-20.06.23*	
2.	*Виконання другого розділу та формування головних показників для розрахунків*	*21.06-14.08.23*	
3.	*Виконання третього розділу, розрахунків та розробка листів*	*15.08-31.10.23*	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	*01.11-10.11.23*	
5.	*Вартісне оцінення ефективності пропозицій роботи*	*20.11-30.11.23*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	*30.11-01.12.23*	
7.	*Завершення роботи в цілому*	*01.12.23-10.01.24*	

Студент _____ Гутиряк А.В.
(підпис)

Керівник роботи _____ Луб П.М.
(підпис)

УДК: 004.75:631.1

Кваліфікаційна робота: 77 с. текст. част., 43 рис., 2 табл., 11 слайдів, 41 джерело.

Інформаційна система моніторингу онлайн-платформ поточного аналізу метео-даних. Гутиряк А.В. Кафедра ІТ. – Дубляни, Львівський НУП, 2024.

Виконано аналіз популярних сайтів прогнозу погоди та онлайн-сервісів спостереження за погодніми умовами із підтримкою API доступу до даних.

Охарактеризовано ТОР глобальних сервісів метеоспостережень із підтримкою API.

Наведено особливості структури даних у форматах JSON та XML, і описано форми відповіді серверів метеосервісів на запити клієнта.

Розроблено алгоритм взаємодії клієнтської частини сервісу із серверною. Наедено структуру сайту та алгоритм його взаємодії із сервером метеосервісу.

Описано причини вибору Framework та інтегровані рішення в інформаційній системі моніторингу онлайн-платформ поточного аналізу метео-даних. Представлено результати розробки діаграми програмної реалізації інформаціної системи та самої інформаційної системи у вигляді веб-орієнтованого додатку.

Наведено результати практичного використання інформаційної системи.

Виконано порівняльну оцінку достовірності онлайн-платформ поточного аналізу метео-даних.

Наведено приклад практичного використання розробки.

Розроблено заходи із охорони праці та безпеки в надзвичайних ситуаціях.

Ключові слова: інформаційна система, метеосервіс, API, веб-додаток, Node Package Manager, Vue, прогнозування, сільське господарство.

ЗМІСТ

ВСТУП

Сьогодні у період активного використання нейронних мереж можна досягнути значної точності прогнозуванян метеорологічних умов для потреб сільськогосподарського виробництва. Водночас, динамічна обробка інформації із метео-станцій спостереження як і будь-які сервіси потребує залучення алгоритмів верифікації та уточнення даних. Саме для таких потреб широко використовують пости наземного спостереження, які в реальному часі передають інформацію на сервери відповідних метеосервісів.

Метеодані та *метеомоніторинг* – це одна з тих складових частин точного землеробства, яка в синергії з іншими дає максимальний результат з наявного у вас ресурсу. *Метеостанція* – це інструмент оцінки стану метеоданих та прийняття рішень. Якщо порівнювати метеомоніторинг з GPS моніторингом, то, як виявилось, найчастіше ефективність застосування такого спостереження починала виявлятися на підприємствах ще до того, як починали аналізувати дані, зібрані з цих пристроїв.

Мета роботи — розробити інформаційну систему що дає змогу отримувати дані з різних глобальних метеосервісів та викнувати метеозведення за назвою населеного пункту.

Завдання роботи: 1) проаналізувати онлайн-сервіси прогнозу погоди та глобальні метеосервіси з підтримкою API доступу до даних; 2) означити структуру даних у форматах JSON та XML; 3) розкрити форми відповіді серверів метеосервісів на запити клієнта; 4) розробити алгоритми взаємодії клієнтської частини сервісу із серверною; 5) описати та розробити веб-орієнтований додаток для відображення прогнозу погоди за даними глобальних метеосервісів; 6) представити результати програмної реалізації інформаційної системи.

Об’єкт роботи – інформаційні сервіси для відображення метеоданих.

Предмет роботи – дані метеомоніторингу та їх відображення у програмній реалізації інформаційної системи.

Новизна одержаних результатів:

- розкрито взаємодію GET-запитів клієнта із обміном даних через API сервера глобального метеосервісу;
- побудовано інформаційну систему та виконано її програмну реалізацію у вигляді веб-орієнтованого додатку;
- отримано та проаналізовано метео-дані з різних глобальних метеосервісів.

Практична цінність роботи – розроблена інформаційна система для представлення і порівняння метеоданих, що включає підтримку API метеосервісів та обмін даними. Обрано інтегроване середовище розробки (Visual Studio Code) та метод створення веб-орієнтованого додатку (Vue.js Framework + Node Package Manager) дало змогу валідно представляти метеодані із глобальних сервісів. Розроблена інформаційна система протестована для різних API метеосервісів.

РОЗДІЛ 1
АНАЛІЗ ПЕРЕДУМОВ ЗАСТОСУВАННЯ МЕТЕОСЕРВІСУ

1.1. Аналіз популярних сайтів прогнозу погоди в Україні

Метеосервіс (або веб-сайт погоди) – тип веб-сайту, який спеціалізується на представленні звітів про поточний стан погоди, а також виконує її прогноз. Всі матеріали, які можуть бути презентовані в подібному сервісі мають різний характер і вид:

- докладні прогнози рівня по годинах на 7 діб вперед;
- узагальнені прогнози по днях на 14 діб вперед;
- короткі прогнози опадів на 2 години вперед з інтервалами по 15 хвилин;
- звіти погоди за місяць і ін.

Метеосервіси з підтримкою інтернет-технологій можуть вирішувати класичні метеорологічні труднощі – зберігання, безпека, забезпечення широкого, швидкого і легкого доступу інформації.

Існують наступні функції метеосервісу:

- інфомаційна;
- культурно-просвітницька;
- навчальна;
- довідкова.

Для того щоб створити онлайн метеосервіс, досить щоб в наявності були необхідне технічне обладнання (комп'ютерна техніка, вихід в Інтернет) і доступ до бази данних метеоцентрів. На сьогоднішній день будь-яка просвітницька установа, а крім того різні компанії, в тому числі і туристичні в нашій державі мають подібне технічне обладнання. Те, що відноситься до співробітників, то є спеціалізовані громадські плани, де навчають роботі з такою технікою. Тут працюють кваліфіковані спеціалісти, який зацікавлений в даному процесі.

Одне з рішень створення метеосервісу базується на відображенні погодних звітів у стилі «плитки» чи структуровно-послідовної інформації у зручному інтерфейсі в різних форматах (погодинний, потижневий та ін.).

Також як додаток до існуючуго функціоналу іноді запроваджують можливість реєстрації та комментарів, аби користувачі могли обговорювати теми, які їх цікавлять, приймати рішення щодо проведення чи скасування якихось заходів та іншого. Але іноді підхід з реєстрацією все ж таки краще залишити, бо навантаження на онлайн ресурс позначається на продуктивності та швидкості завантаження контенту на сторінки. Існує багато методів боротьби з покращення продуктивності: один з кращих – обирати бізнес-логіку та інструменти розробки згідно з метою розробки (рис. 1.1) [18].

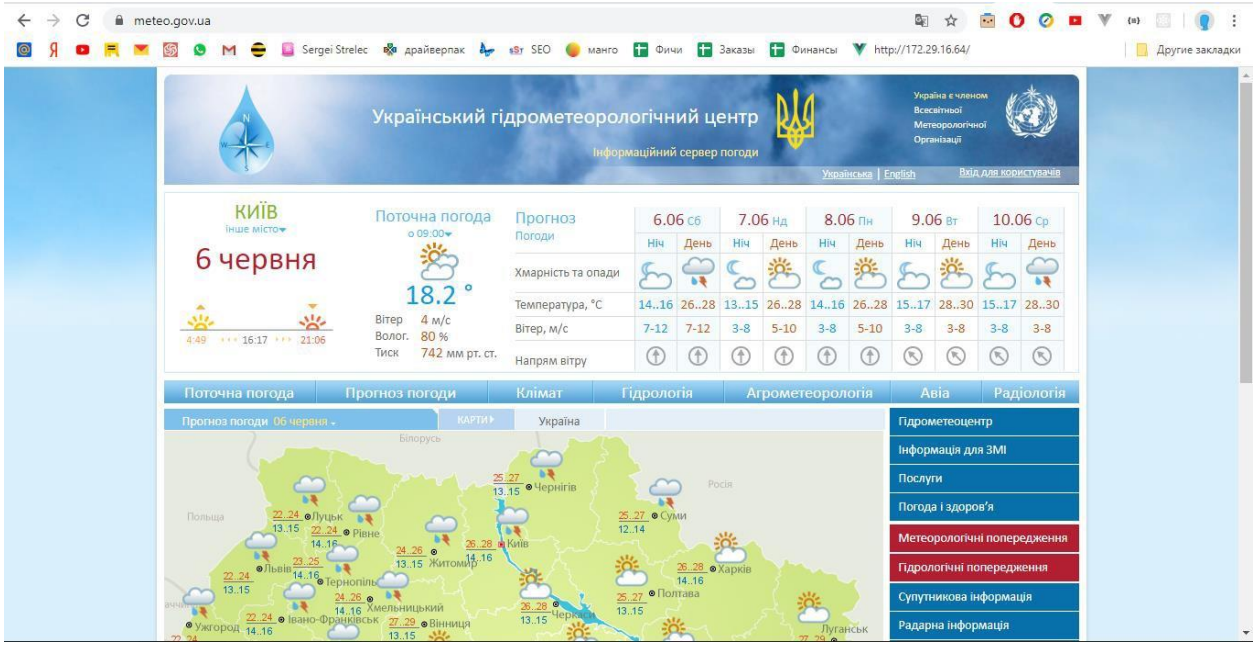


Рисунок 1.1 – Гідрометеорологічний центр

Також деякі метеосервіси демонструють інформацію по погоді в вигляді динамічної карти з різних кутів зору та описом параметрів звіту погоди (рис. 1.2 та рис. 1.3.) [19].

Також існує метеорологічна компанія з назвою «The Ventusky». Чеські розробники створили високоякісний додаток, який чітко відображає метеорологічні дані з усього світу та дозволяє стежити за розвитком погоди в будь-якому місці Землі.

Рисунок 1.2 – Синоптик із динамічними даинми [19]

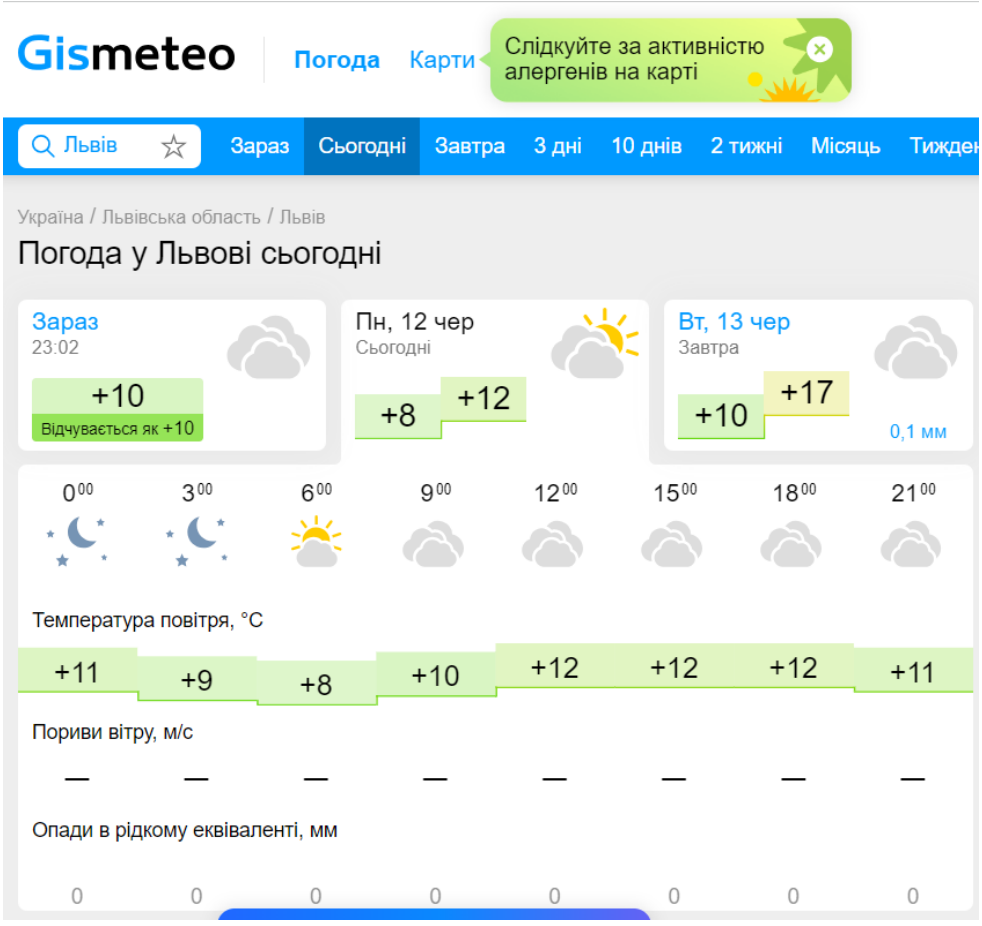


Рисунок 1.3 – Гісметео із динамічними даними [18]

Погода Землі функціонує як взаємозалежна система.

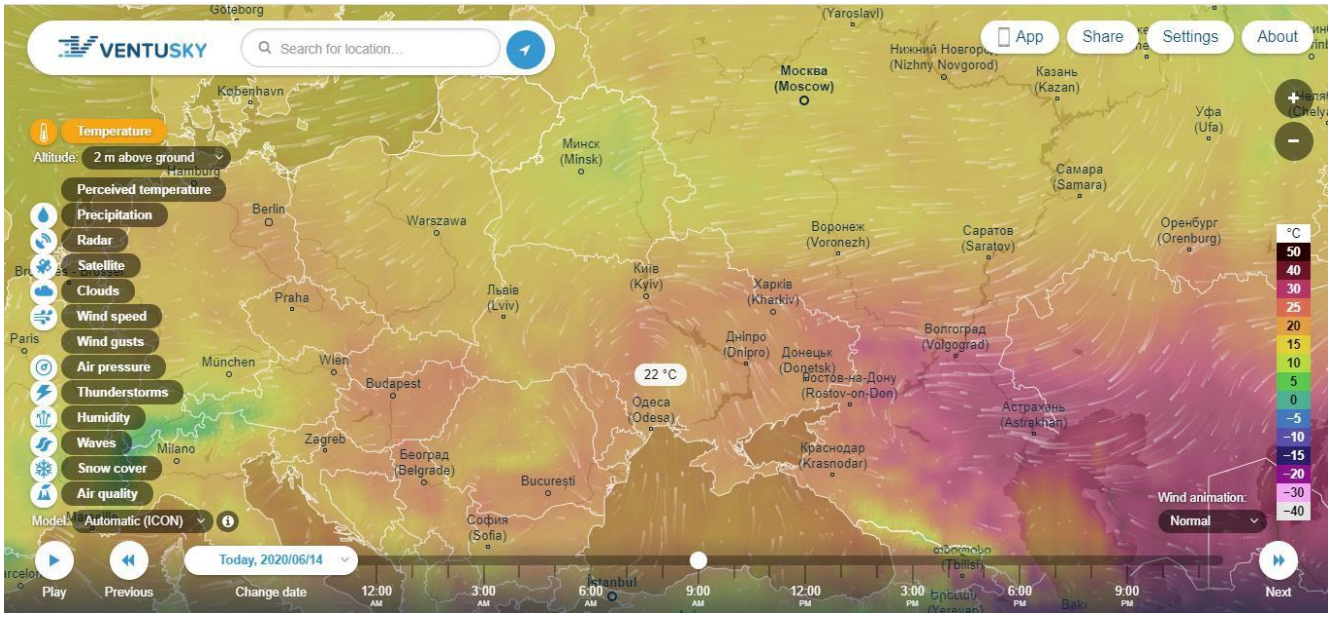


Рисунок 1.4 – Метеосервіс «The Ventusky»

«The Ventusky» створили абсолютно нову систему відображення хвиль. За допомогою використання анімованих дуг візуалізація чітко розмежовує напрямок руху та висоту як вітрових хвиль, так і напрямків. Для інших метеорологічних елементів було обрано кольорові шкали, які належним чином ілюструють опади, тиск повітря та температуру.

Додаток Ventusky доступний для всіх у всьому світі, спрямований на покращення поінформованості про метеорологічні події в нашій атмосфері (рис. 1.4).

1.2. Аналіз онлайн-сервісів спостереження за погодніми умовами із підтримкою API доступу до даних

Метеосервіси присвячені наданню глобального прогнозу погоди та змісту погоди для веб-сайтів, підприємств та туристичної галузі. Про їх створенні мріяли ще на зорі інтернету, але тільки в останні роки завдяки вдосконаленню технологій стали з'являтися красиві і інформативні ресурси.

Поняття «метеосервіс» містить в собі збірку різного роду Web-сторінок. Вони повинні розташовуватися на одному або декількох Web-серверах. Подібні сторінки містять в собі систематизовану інформацію метео-даних [3, 23].

Сервіс ***AccuWeather*** – це американська медіа-компанія, що надає комерційні послуги з прогнозування погоди по всьому світу. Данний сервіс надає прогнози та попередження про погоду, а також додаткові продукти та послуги, пов'язані з погодою, клієнтам по всьому світу у ЗМІ, бізнесі та уряді, включаючи більше половини компаній зі списку Fortune 500 та тисячі інших компаній у всьому світі.

Сервіс славиться високою точністю прогнозів погоди. Служби прогнозів та попереджень базуються на інформації про погоду, отриману з численних джерел, включаючи спостереження за погодою та дані, зібрані Національною метеорологічною службою та метеорологічними організаціями за межами США, а також на інформації, наданій не метеорологічними організаціями, такими як Агентство з охорони навколишнього середовища та збройних сил, а також за власними моделями та алгоритмами.

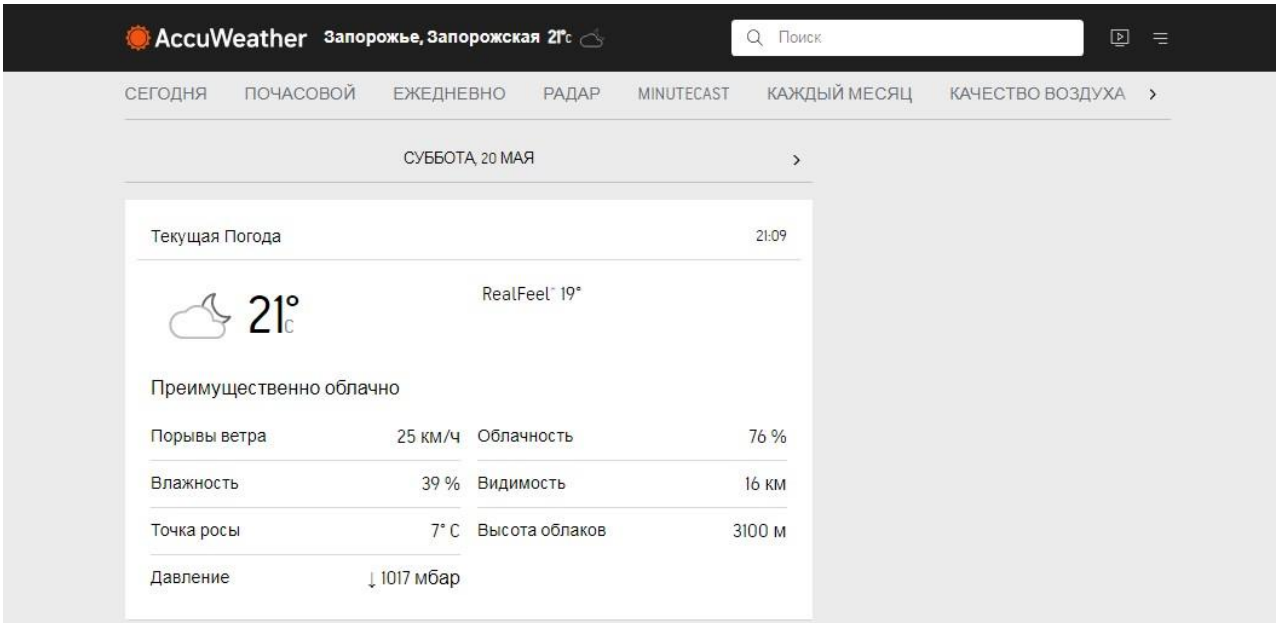


Рисунок 1.5 − Інтерфейс сервісу AccuWeather

Надає прогнози на різні періоди часу, включаючи поточну погоду, прогноз на годину вперед, прогноз на кілька днів вперед і навіть довгострокові прогнози, і навіть на кожен місяць. Це дозволяє користувачам отримувати інформацію про

погоду на потрібний проміжок часу. Має широкий набір погодних параметрів, включаючи температуру, опади, вологість, швидкість та напрям вітру, тиск, індекси рівня комфорту та багато іншого.

Для розробки веб-додатків або програм, є можливість отримувати дані про погоду за допомогою API у таких форматах як XML та JSON. Сервіс AccuWeather має як безкоштовний так і платний доступ. У безкоштовному плані є доступ тільки до базових функцій та обмеженої кількості запитів на день. У платному плані є все те, що було і у безкоштовному, але з додатковими можливостями для бізнес-клієнтів або користувачів із підвищеними вимогами.

Сервіс ***OpenWeatherMap*** − це популярний онлайн сервіс, який використовує API для надання прогнозу на поточний, годинний, майбутній період та надання історичних даних про погоду.

Сервіс намагається забезпечити високу точність прогнозів погоди, використовуючи в якості джерел, офіційні метеорологічні служби, дані з метеостанцій аеропортів та дані з приватних метеостанцій, щоб надати актуальну та достовірну інформацію про погоду. Однак, точність може змінюватись в залежності від місця розташування та часу.

Отримана інформація оброблюється за допомогою сервіса OpenWeatherMap, після чого, на основі даних будується прогноз погоди, карти хмарності, опадів, погодні карти, тощо. Основною ідеєю даного сервісу є використання приватних погодних станцій, які допомагають підвищити точність вихідної погодної інформації та, як наслідок, точність прогнозів погоди.

OpenWeatherMap надає розробникам доступ до свого API, що дозволяє інтегрувати прогнози погоди та метеорологічні дані у власні веб- програми, мобільні програми або інші системи. API цього сервісу надає різні методи та параметри, що дозволяють отримувати необхідну інформацію про погоду. OpenWatherMap використовує платний API, але є і безкоштовний функціонал проте він обмежений у використанні. Всі погодні дані можна отримати у форматах JSON, XML або HTML.

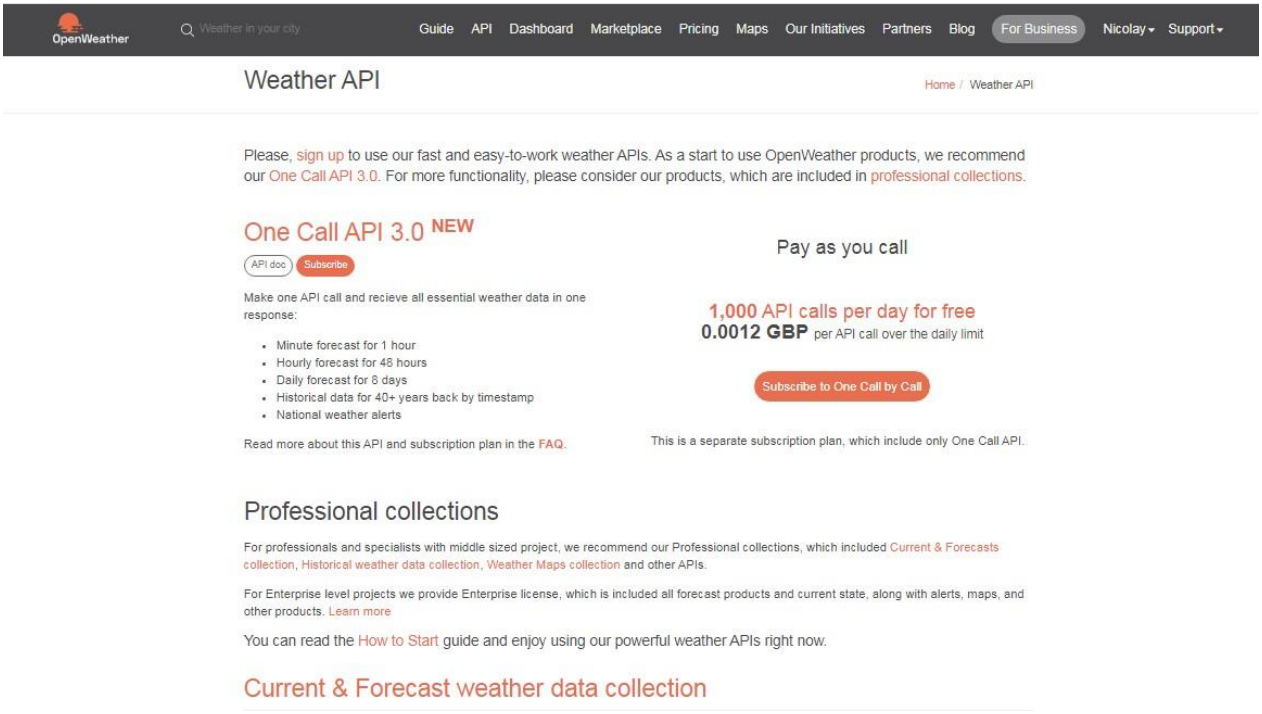


Рисунок 1.6 − Інтерфейс сервісу OpenWeatherMap

Сервіс ***Weatherbit*** – це глобальний сервіс прогнозу погоди, який призначений для надання максимально якісних прогнозів погоди, спостережень та історичних даних про погоду.

Система пронозування Weatherbit використовує глобальні прогностичні моделі такі як − GFS / ECMWF, у поєднанні з локальними моделями ближньої дії з високою роздільною здатністю, такими як моделі HRRR , NAM і DWD ICON для отримання максимально точних прогнозних даних.

Даний сервіс включає в себе найкращі у світі джерела даних про погоду упаковуючи їх у простий та зручний формат, такий як JSON.

Сервіс при собі має широкий спектр пошуку погоди і за допомогою API ключа, який надається цим сервісом. Можна дізнатися прогноз погоди на поточний час, прогноз погоди на 16 днів, почасовий прогноз на 240 годин/щогодини, прогноз опадів погодинно, якість повітря, температуру охолодження або опалення. Також надається інформація про різні погодні параметри, такі як температура, вологість, швидкість вітру, опади, хмарність, тиск та інші.

Дані про прогноз погоди, беруться зі супутників, радарів, аналізу різних

метеостанцій за допомогою машинного навчання, тощо.

Рисунок 1.7 − Інтерфейс сервісу Weatherbit

Сервіс ***Meteomatics*** – це сервіс прогнозу погоди, який спеціалізується на комерційному прогнозуванні прогнозу погоди з високою роздільною здатністю, прогнозуванні потужності сонця, вітру, гідроелектростанцій та збирання даних про погоду з нижніх шарів атмосфери за допомогою метеородронів.

Сервіс Meteomatics, надає високоякісну інформацію про погоду дозволяючи клієнтам щодня приймати більш виважені рішення. Від підвищення безпеки на літаках та кораблях до економії палива та оптимізації роботи об'єктів відновлюваної енергетики. Доступ до точних даних про погоду допомагає компаніям підвищувати продуктивність та знижувати ризики для свого бізнесу.

Завдяки спеціальному дозволу Федерального управління цивільної авіації, метеорологічні дрони можуть літати за межами прямої видимості BVLOS, що дозволяє збирати дані навіть у хмарах та тумані. Meteomatics інтегрує дані, зібрані дронами, для забезпечення, ще більш точного та надійного прогнозу погоди.

Для розробки веб-додаків передбачено можливість отримати дані прогнозу погоди через API у різних форматах, таких як : JSON та CSV.

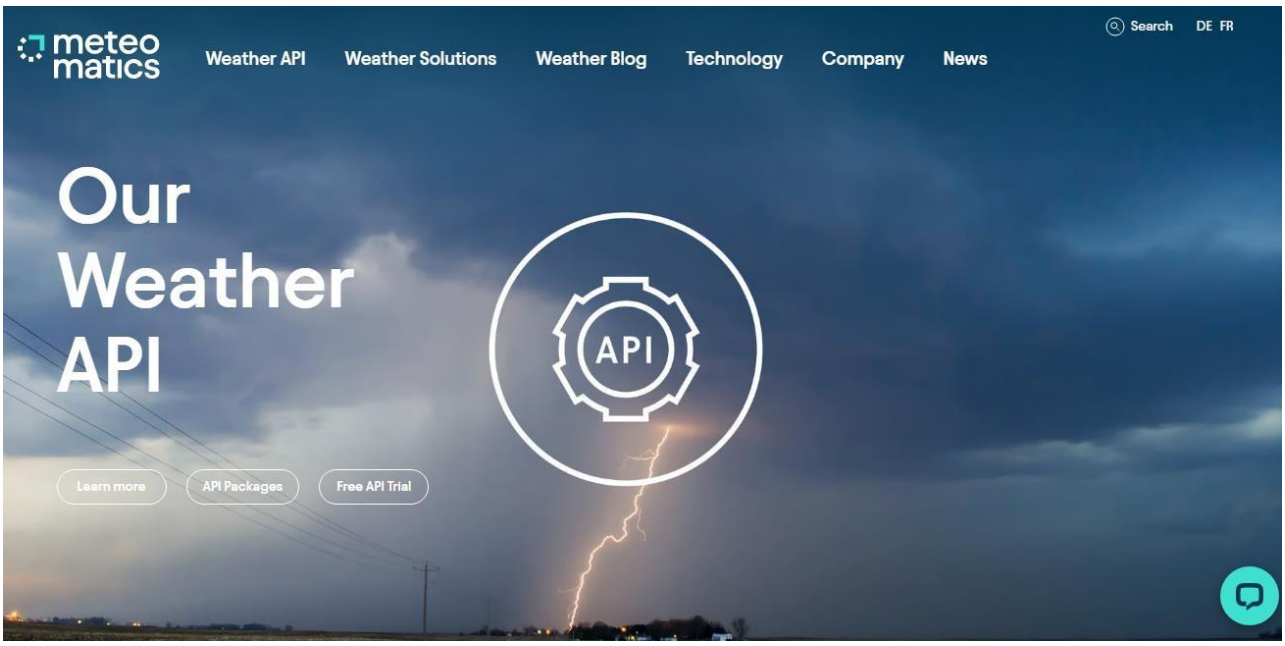


Рисунок 1.8 – Інтерфейс сервісу Meteomatics

Сервіс ***WeatherAPI*** – сервіс прогнозу погоди, який надає прогноз погоди незважаючи на місцезнаходження користувача. Отримання даних про погоду надходить з різних джерел, супутників метеорологічних станцій тощо. Отримані дані оброблюються даним сервісом, щоб надати користувачу дані про погоду. Але погода може трохи відрізнятися від тієї, що показується на сервісі, це може бути пов'язано з поточним місцезнаходженням, але сервіс WeatherAPI надає точні прогнози.

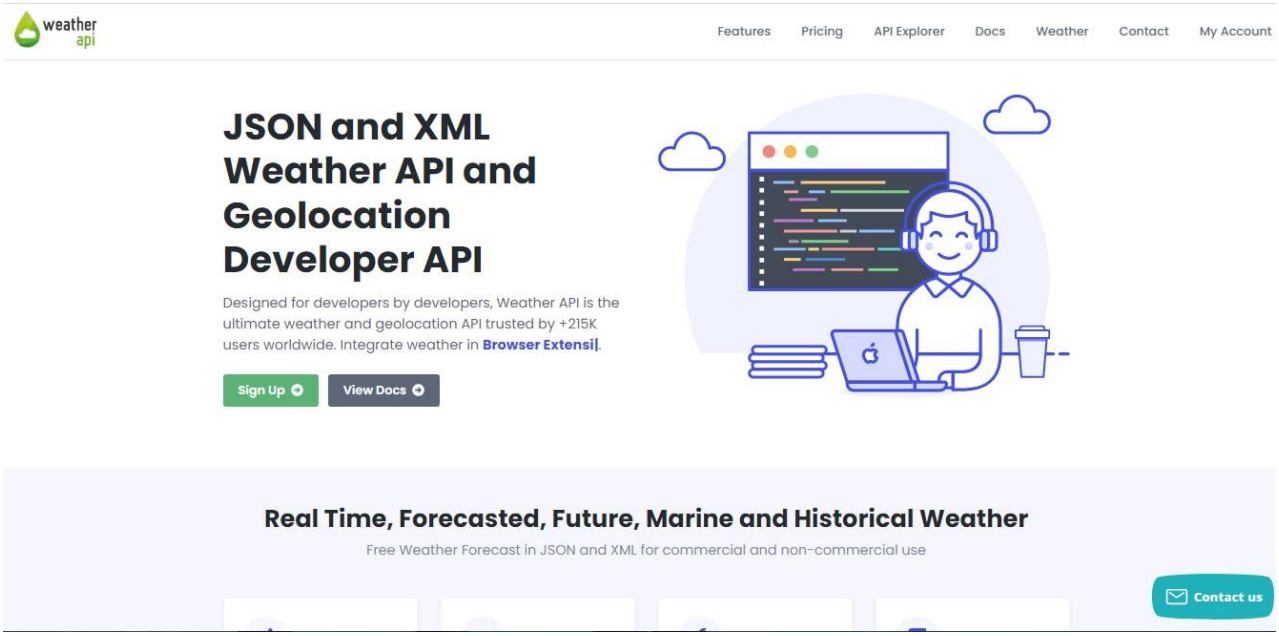


Рисунок 1.9 – Інтерфейс сервісу WeatherAPI

Отримання та відображення даних про прогноз погоди у свої програмах або веб-додатках, відбувається за допомогою API ключа у форматах JSON, XML. Також API може надати інформацію про часовий пояс, астрономічні дані та дані про географічне розташування.

В цьому сервісі є безкоштовний, так і платний тариф. Відображення прогнозу погоди у безкоштовному тарифі, можливе на поточний час і тільки на 3 дні. У платних тарифах, є можливість відобразити прогноз погоди на 5, 7, 14 днів. Але при реєстрації дається пробний тариф, терміном дії 30 днів, який може відобразити прогноз погоди на 14 днів.

Таким чином, кожний популярний онлайн метео-сервіс надає послуги доступу до власного API та бази данних звітів погоди та геоположень. Ця послуга буває безкоштовна, але чаще всього щоб отримати доступ до API треба оформити платну підписку та отримати ліцензований ключ доступу. Розробник має можливість ввести цей ключ до параметрів зверення запиту та отримати повний список інформації погоди у різних форматах:

- XML;
- CSV;
- JSON.

1.3. TOP глобальних сервісів метеоспостережень із підтримкою API

Сьогодні є доступними глобальні онлайн-сервіси, які дозволяють стежити та прогнозувати погодні, метеорологічні, кліматичні умови та явища. Їх існує чимало і загалом дають змогу вирішувати різні задачі, зокрема, ***OSINT/GEOINT***.

Розвідка на основі відкритих джерел (англ. *Open source intelligence,* ***OSINT***) — концепція, методологія і технологія добування і використання військової, політичної, економічної та іншої інформації з відкритих джерел, без порушення законів. Використовується для прийняття рішень у сфері національної оборони та безпеки, розслідувань тощо.

Геопросторова розвідка (ГПР), в англомовних джерелах відома під скороченнями ***GEOINT***, GeoIntel або GSI, є розвідувальною діяльністю, що полягає в дослідженні та аналізі зображень і геопросторових даних, в результаті яких описуються, оцінюються і візуалізуються фізичні характеристики та географічно локалізовані процеси на Земній кулі. Компонентами ГПР є зображення, оптична (видова) розвідка і геопросторова інформація.

Отже, проаналізуємо найвідоміші глобальні сервіси метеоспостережень із підтримкою API.

Windy.com – це один з кращих онлайн-сервісів для прогнозу погодних умов. Це не просто «погодник», а цілий онлайн-метеоцентр з погодинними подобовими/ щотижневими/ щомісячними прогнозами погоди, радарами, супутниками, мапами вітрів та магнітних хвиль й іншими корисними фішками. Сервіс зручний тим, що дозволяє здійснювати постійний моніторинг за вибраними місцями з автоматичними оповіщеннями на електронну пошту.

Windy може збирати численні аналітичні дані, зокрема: 1) сила і напрям вітру; 2) стеження за пересуванням хмар; 3) температурна карта; 4) опади та атмосферні явища: сніг, дощ, блискавка, шторми, урагани і т.д.; 5) атмосферний тиск; 6) магнітні бурі; 7) якість повітря; 8) точка роси; 9) індекс CAPE; 10) концентрація CO.

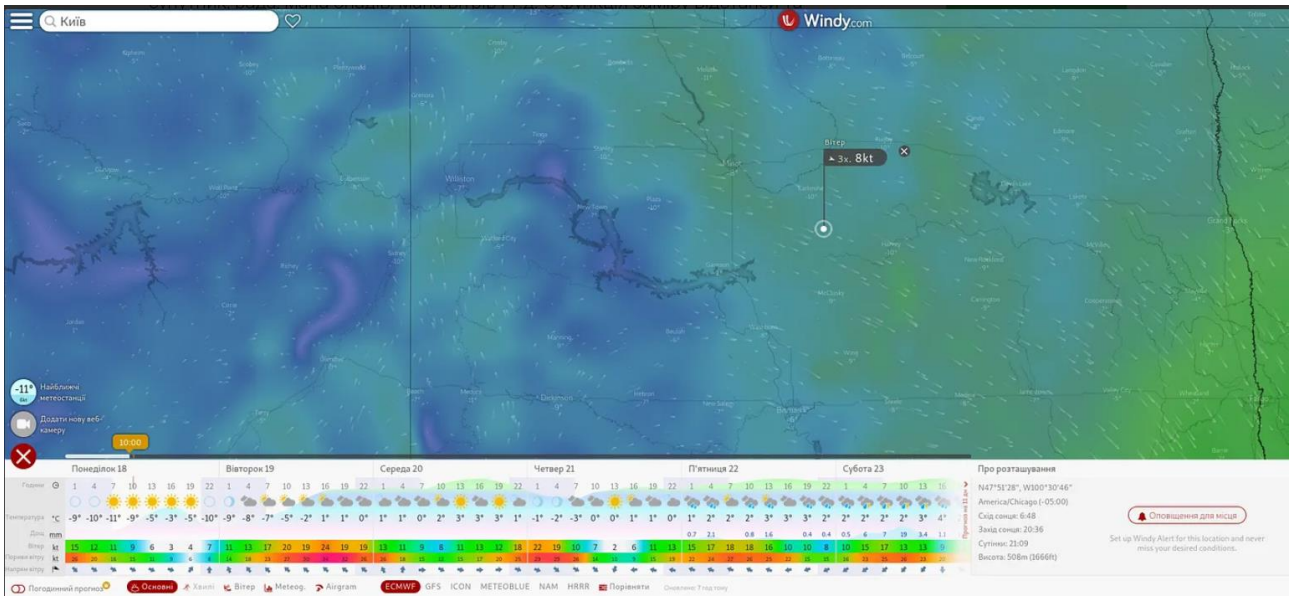


Рисунок 1.10 − Інтерфейс сервісу Windy

З додаткових можливостей слід виділити також підтримку API, можливість відображення поточної ситуації в аеропортах, показ місць, придатних для дельтапланеризму, відображення онлайн-камер і таке інше. Окрім всього, Windy має вбудований мікросервіс веб-камер, які можна знаходити та переглядати по всьому світу:

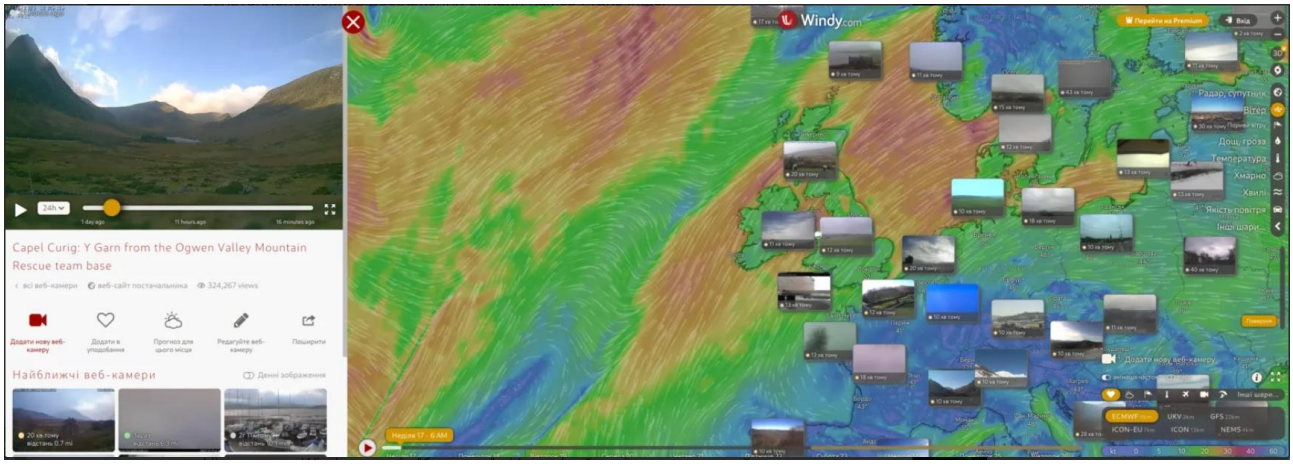


Рисунок 1.11 − Інтерфейс мікросервісу веб-камер Windy

WunderMap − простенький інтерактивний сервіс для спостереження за погодними умовами. Збирає базові метрики (температура, опади, вітри, вологість, забруднення повітря та ін.) і відображає все це на онлайн-мапі. Підтримує різні режими відображення: радар, сателіт, атмосферні явища. Дозволяє експортувати усі налаштування в файл. З недоліків—довго завантажується і рендерить нові шари.

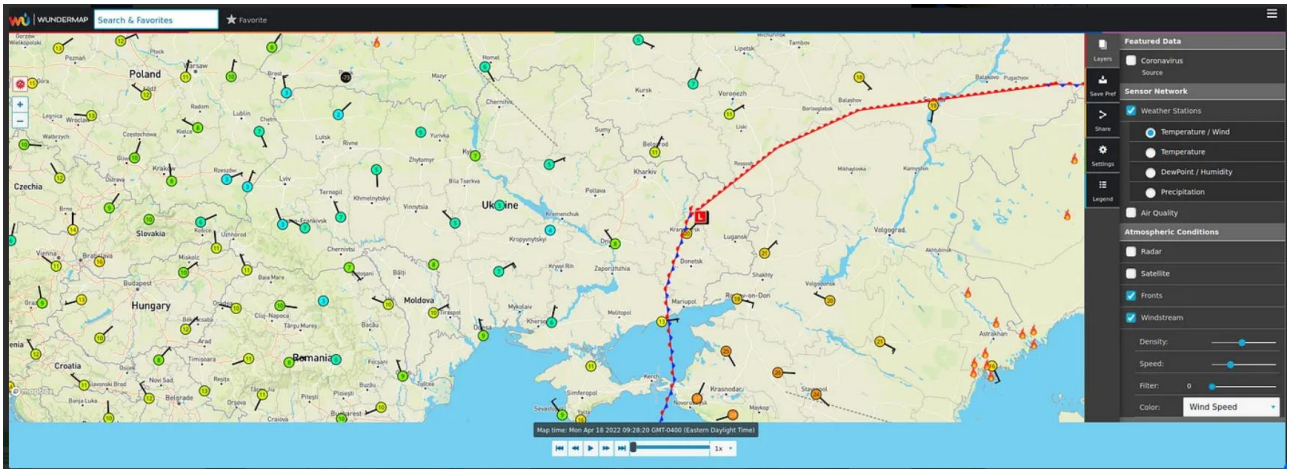


Рисунок 1.12 − Інтерфейс сервісу WunderMap

ZoomEarth – це сервіс у вигляді онлайн-мапи з численними радарами, відображенням погодних умов, атмосферними явищами у реальному часі з використанням анімації.

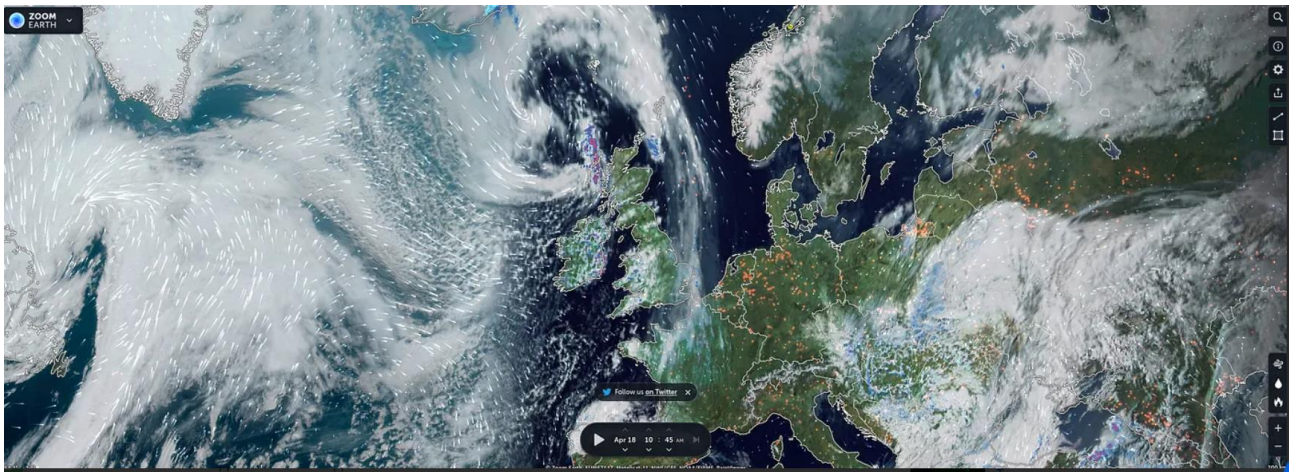


Рисунок 1.13 − Інтерфейс сервісу ZoomEarth

Прогнози погоди оновлюються кожні 10–15 хвилин з геостаціонарних супутників: NOAA GOES, NASA-NOAA, JMA Himawari-8, EUMETSAT Meteosat. Треки тропічних циклонів і прогнозні карти створені з використанням останніх даних від NHC, JTWC, NRL та IBTrACS. Анімовані мапи прогнозу швидкості приземного вітру створюються за допомогою моделі NOAA-NWS GFS. Радарні карти дощу надаються RainViewer. Теплові карти показують місця пожеж та джерел високої температури з використанням даних FIRMS та InciWeb.

Meteoblue – онлайн-сервіс з поглибленими метеорологічними властивостями. Корисний для завдань метеорології або прогнозування погодних явищ.

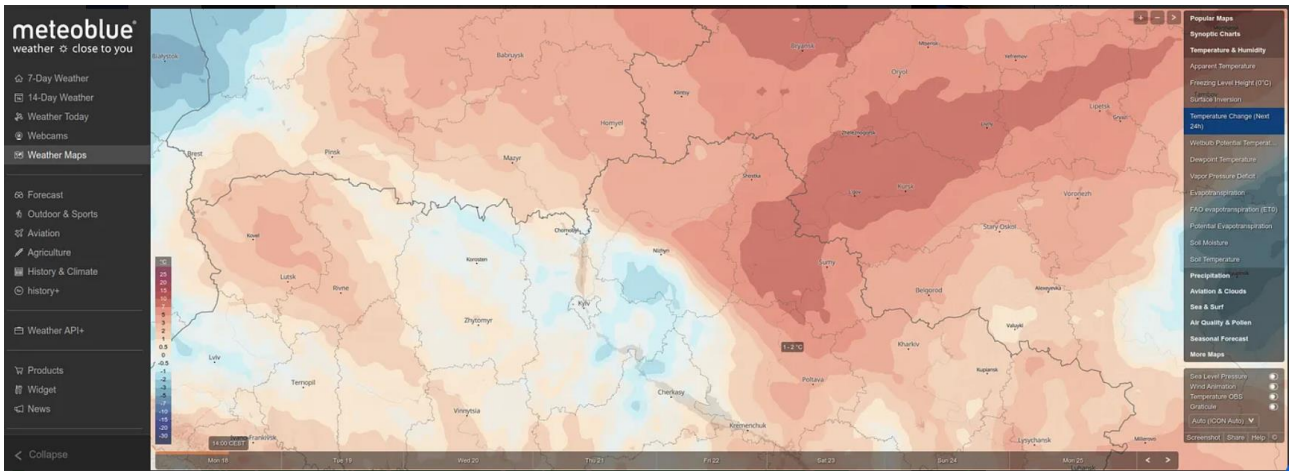


Рисунок 1.14 − Інтерфейс сервісу Meteoblue

Онлайн-мапа MeteoBlue підтримує величезну кількість шарів накладання: хмари, температура, опади, море, повітря, сонячна активність і т.д. Кожен шар володіє численними режимами і характеристиками відображення.

LightingMaps.org – вервіс LightingMaps показує на мапі дані про грози і блискавки (удари, розряди) в різних частинках земної кулі.

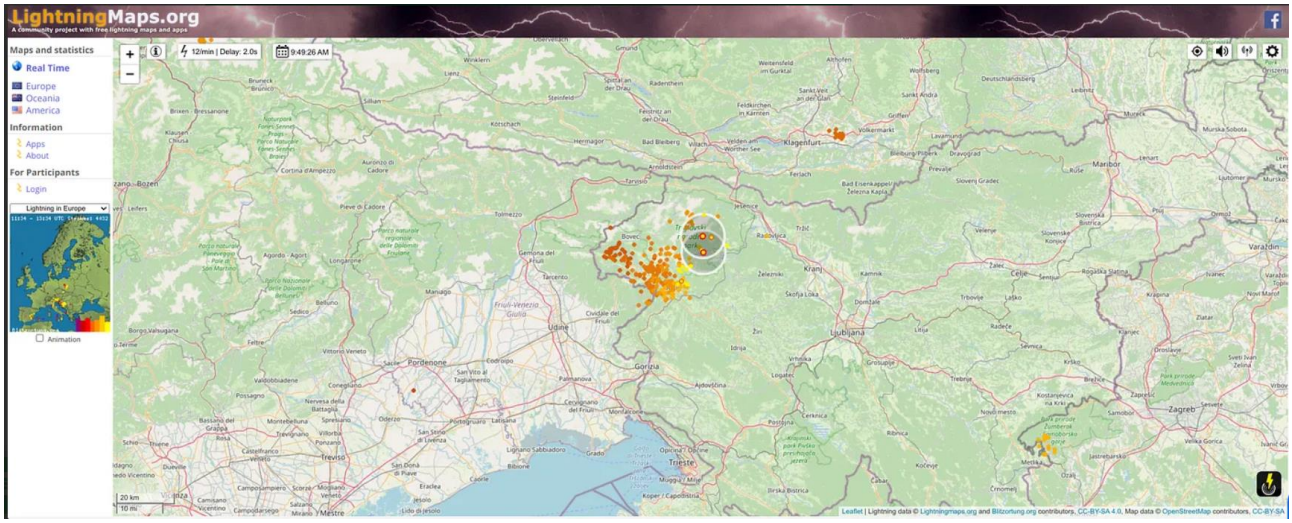


Рисунок 1.15 − Інтерфейс сервісу LightingMaps

Є частиною проєкту https://www.blitzortung.org/—спільнота дослідників атмосферних явищ.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЗАЄМОДІЇ КЛІЄНТА ІЗ СЕРВЕРОМ ОНЛАЙН-СЕРВІСУ ПОГОДИ ЗА ДОПОМОГОЮ API

2.1. Структура даних у форматах JSON та XML

JSON (*JavaScript Object Notation*) – це формат обміну даними, який легко читати та писати людям, а машинам легко аналізувати та генерувати. Він заснований на підмножині мови програмування JavaScript, стандарт ECMA-262, JSON — це текстовий формат, який повністю не залежить від конкретної мови програмування, але використовує умови, які знайомі програмістам мов С-родини, включаючи C , C++ , C#, Java, JavaScript, Perl, Python та багато інших. Ці властивості роблять JSON ідеальним форматом для обміну даними між різними мовами програмування.

JSON використовує простий та інтуїтивно зрозумілий синтаксис, що базується на JavaScript. Він складається з пар "ключ-значення", де ключі представлені у вигляді рядків, а значення можуть бути рядками, числами, значеннями булевими, масивами, об'єктами або null.

Структура даних JSON включає в себе Об’єкти та Масиви.

```
{
  "orderID": 12345,
  "shopperName": "Ivan Ivanov",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Super product",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Wonderful product",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Рисунок 2.1 − Приклад схеми “ключ – значення”

Об'єкти: JSON дозволяє створювати невпорядковані колекції пар “ключ-значення” (рис. 2.1). Об'єкти пишуться у фігурні дужки {}, і кожна пара "ключ-значення" розділяється комою. Ключі подаються у вигляді рядків, а значення може бути будь-якого типу даних, що підтримується.

Масиви: JSON підтримує впорядковані списки значень. Масиви пишуться у квадратних дужках [], і елементи поділяються комами. Значення у масиві може бути будь-якого типу даних.

```
["apple", "banana", "orange", 41, 3.14, true, false, null]
```

Рисунок 2.2 – Підтримка різних типів даних у масиві

Також формат JSON підтримує такі *типи даних*:

– рядки. Рядки подаються у вигляді послідовності символів, які записуються у подвійні лапки ` “ ” `.

– числа. Числа в JSON можуть бути цілими чи числами з плаваючою точкою “ 41” або “3.14”.

– булеві значення. JSON підтримує логічні значення true та false.

– Null. Має спеціальне значення null, яке представляє порожнє чи відсутність значення.

Також JSON дозволяє додавати поля користувача і розширювати дані без зміни структури всього документа. Це забезпечує гнучкість під час роботи з даними. Але він не підтримує коментарів у синтаксисі. Весь текст у документі JSON розглядається як дані.

Коли користувач відправив GET-запит на потрібний сервіс прогнозупогоди. Він отримує відповідь у певному форматі. Відображення отриманих даних від сервера у форматі JSON :

GET-запит

{
"coord": {
"lon": 10.99,
"lat": 44.34
},
"weather": [

{
"id": 501,
"main": "Rain",
"description": "moderate rain","icon": "10d"
}
],
"base": "stations","main": {
"temp": 298.48,
"feels_like": 298.74,
"temp_min": 297.56,
"temp_max": 300.05,
"pressure": 1015,
"humidity": 64,
"sea_level": 1015,
"grnd_level": 933
},
"visibility": 10000,"wind": {
"speed": 0.62,
"deg": 349,
"gust": 1.18
},
"rain": {
"1h": 3.16
},
"clouds": {
"all": 100
},
"dt": 1661870592,
"sys": {
"type": 2,
"id": 2075663,
"country": "IT", "sunrise": 1661834187,
"sunset": 1661882248
},
"timezone": 7200,
"id": 3163858,
"name": "Zocca","cod": 200

Так як такий формат дуже гнучкий, то він широко використовується для обміну даними між клієнтськими та серверними програмами, веб-сервісами, програмним інтерфейсом API, зберігання конфігураційних даних та серіалізації об'єктів. Він також часто використовується в різних областях, включаючи веб-розробку, мобільні програми та хмарні сервіси. Тому JSON має простий і зрозумілий синтаксис, компактне представлення даних і широку підтримку в різних мовах програмування. Це робить його зручним форматом для обміну та зберігання структурованих даних.

Формат XML. XML (eXtensible Markup Language) – це мова розмітки, яка рекомендована Консорціумом Всесвітньої павутини (W3C). Формат XML описує XML-документи та частково описує поведінку XML-процесорів.

XML розроблявся як мова з простим формальним синтаксисом, яка буде зручною для обробки та створення документів як людиною, так і програмами, з акцентом на використання в Інтернеті. Мова називається розширюваною, оскільки вона не фіксує розмітку, що використовується в документах, а розробник сам може створити свою розмітку відповідно до потреб і до конкретної області, яку він вибрав, будучи обмеженим лише в синтаксичному правилі написання мовою XML.

Розширення XML – це конкретна граматика, яка була створена на базі XML і являє собою набір словникових тегів та їх атрибутів, а також набором правил, що визначають, які елементи та атрибути можуть входити до складу інших елементів. XML поєднує в собі простий формальний синтаксис, який буде зручний і зрозумілий для людини, а також сам формат базується на кодуваннях Юнікод для представлення змісту документів. Це призвело до широкого використання, як власне XML, так і безлічі похідних спеціалізованих мов на базі XML або на найрізноманітніших програмних засобах.

Елемент у XML – це логічна структура документа. Кожен документ містить один або кілька елементів (рис. 2.3).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE recipe>
3  <recipe name="Тісто" preptime="5min" cooktime="180min">
4      <title>
5          Приготування тіста
6      </title>
7      <composition>
8          <ingredient amount="3" unit="стакан">Мука</ingredient>
9          <ingredient amount="0.25" unit="грамм">Дріжжі</ingredient>
10         <ingredient amount="1.5" unit="стакан">Тепла вода</ingredient>
11     </composition>
```

Рисунок 2.3 – Розширення та елементи XML

Границі елементів представлені початковим та кінцевим тегами. Ім'я елемента в початковому та кінцевому тегах елемента має співпадати. Елемент також може бути представлений, як тегом порожнього елемента, тобто не

включає інші елементи і символьні дані. Елементи можуть містити текстовий вміст, який представляє фактичні дані. Текстовий вміст знаходиться між початковим та закриваючим тегами елемента.

Відображення отриманих даних від сервера у форматі XML :

GET-запит

```
<current>
<city id="3163858" name="Zocca">
<coord lon="10.99" lat="44.34"/>
<country>IT</country>
<timezone>7200</timezone>
<sun rise="2022-08-30T04:36:27" set="2022-08-30T17:57:28"/>
</city>
<temperature value="298.48" min="297.56" max="300.05" unit="kelvin"/>
<feels_like value="298.74" unit="kelvin"/>
<humidity value="64" unit="%"/>
<pressure value="1015" unit="hPa"/>
<wind>
<speed value="0.62" unit="m/s" name="Calm"/>
<gusts value="1.18"/>
<direction value="349" code="N" name="North"/>
</wind>
<clouds value="100" name="overcast clouds"/>
<visibility value="10000"/>
<precipitation value="3.37" mode="rain" unit="1h"/>
<weather number="501" value="moderate rain" icon="10d"/>
<lastupdate value="2022-08-30T14:45:57"/>
</current>
```

Тому, XML є дуже хорошим форматом, бо він універсальність, гнучкість і незалежність від платформи, що робить його широко застосованим форматом для обміну даними, файлів конфігурації, веб-сервісів та інших додатків, де потрібне представлення та обмін структурованими даними.

2.2. Алгоритм взаємодії клієнтської частини сервісу із серверною

API – це такі механізми, які дозволяють двом програмним компонентам взаємодіяти один з одним, використовуючи набір визначень та протоколів. Наприклад, система метеослужби містить щоденні і майбутні дані про погоду. Програма погоди на телефоні чи на комп’ютері після запиту спілкується з цією

системою (сервісом) через API і показує щоденні, або майбутні оновлення погоди на комп'ютері чи на телефоні.

Також API – *Application Programming Interface*, що означає програмний інтерфейс програми. У контексті API слово «додаток» стосується будь-якого програмного забезпечення з певною функцією. Інтерфейс можна розглядати як сервісний контракт (як посередник) між двома програмами. Цей контракт (посередник) визначає, як йде взаємодія один з одним, використовуючи запити і відповіді. У документаціях по API міститься інформація про те, як розробники правильно повинні структурувати ці запитита отримувати відповідні відповіді.

Робота API зазвичай пояснюється з погляду клієнта та сервера. Програма або веб-додаток, яка надсилає запит на сервер, називається клієнтом, а програма, що надсилає відповідь клієнту, називається сервером. Тому на прикладі з погодою, база даних сервісу прогнозу погоди – це сервер, а програма або веб-додаток – це клієнт, який отримує дані від сервера.

REST – це *Representational State Transfer*, тобто передача репрезентативного стану. REST визначає набір певних функцій, таких як GET, PUT, DELETE тощо, які клієнти можуть використовувати для доступу до даних сервера. Клієнти та сервери обмінюються даними за протоколом HTTP.

У моєму випадку йде використання лише функції GET запиту, тому що ми тільки отримуємо дані та відображаємо їх на своєму веб-сервісі.

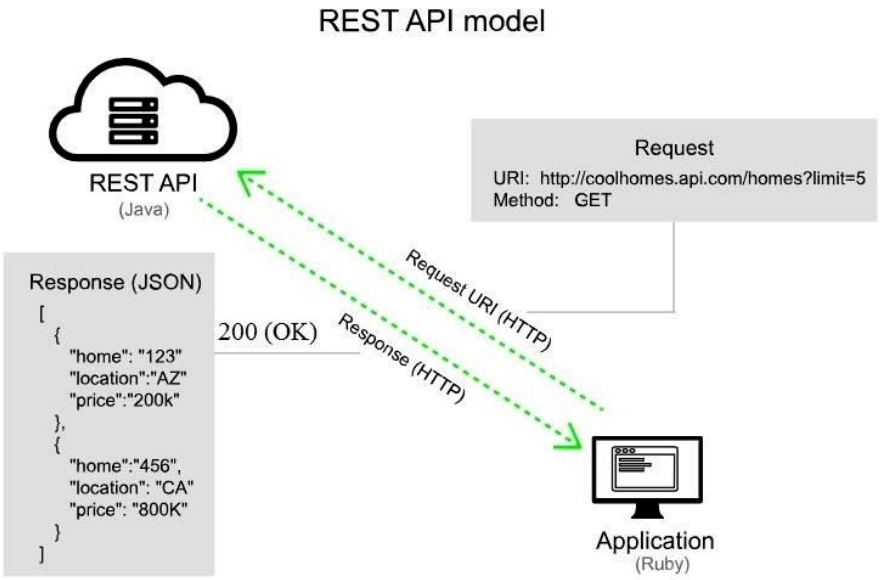


Рисунок 2.4 − Схема алгоритму GET-запиту клієнта до сервера

Головною особливістю REST API є те, що така передача виконується без збереження стану. Це означає, що сервери не зберігають дані клієнта між запитами. Запити клієнтів до сервера подібні до URL-адрес, які вводяться в браузері при відвідуванні веб-сайту. Відповідь від сервера є простими даними без типового графічного відображення веб-сторінки.

На сьогоднішній день REST API є найпопулярнішим і гнучким API-інтерфейсом в Інтернеті. Клієнт надсилає запити на сервер у вигляді даних. Сервер використовує це введення клієнта для запуску внутрішніх функцій і повертає вихідні дані назад клієнту.

Рисунок 2.5 − Приклад взаємодії клієнта із сервером за допомогою API

До REST API можна віднести чотири переваги :

− *інтеграція*. API використовуються для інтеграції нових програм із існуючими програмними системами. Це дає змогу збільшити швидкість розробки, тому що кожну функцію не потрібно писати з нуля.

− *інновації*. Цілі галузі можуть змінитися з появою нової програми. Компанії повинні швидко реагувати та підтримувати швидке розгортання інноваційних послуг. Вони можуть це зробити, вносячи зміни на рівні API без необхідності переписувати весь код.

− *розширення*. API-інтерфейси надають компаніям унікальну можливість задовольняти потреби своїх клієнтів на різних платформах. Наприклад, карти API дозволяє інтегрувати інформацію про карти через веб-сайти, iOS, Android тощо. Будь-яка компанія може надати аналогічний доступ до своїх внутрішніх баз даних, використовуючи безкоштовні або платні API.

− *простота обслуговування.* API діє як шлюз між двома системами. Кожна система зобов'язана вносити деякі внутрішні зміни, щоб це не вплинуло на API. Таким чином, будь-які майбутні зміни в самому коді однією стороною не повині вплинути на іншу сторону.

2.3. Структура сайту та алгоритм його взаємодії із сервером метеосервісу

Структура сайту – це логічна побудова всіх сторінок сайту, категорій та підкатегорій. Це логічна схема, відповідно до якої всі сторінки та розділи сайту розташовані відносно один одного та принцип, за яким вони взаємопов'язані один з одним. З технічної точки зору, навігація ресурсу є набір URL, логічно збудованих у певній послідовності.

Найбільш популярна схема структури сайту це деревоподібна структура.

Деревоподібна структура – універсальна структура, що підходить для сайтів різних типів. Ця структура – ієрархічна, вона містить розділи, які у свою чергу включають кілька підрозділів і так далі. Таким чином, різні сторінки сайту мають різні рівні вкладеності.

На рисунку 2.6 представлена схема структури сайту.

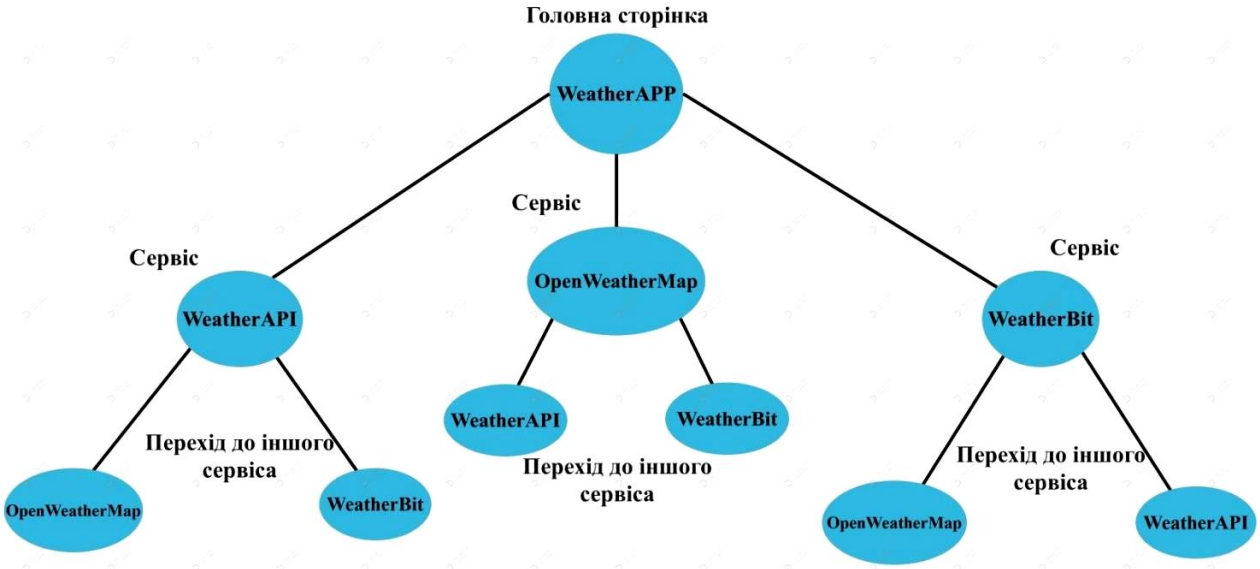


Рисунок 2.6 − Схема структури сайту

Правильність, простота та логічність структури сайту безпосередньо впливає на працездатність сайту. Від того, наскільки проста та зрозуміла структура сайту, залежить те, як швидко відвідувач знайде потрібну йому інформацію.

Алгоритм взаємодії веб-додатку з сервером прогнозу погоди. Клієнт у веб-додаток формує GET-запит, вказуючи необхідну URL-адресу сервера погоди, API ключ, а також місто або його координати. Після чого цей GET-запит відправляється на сервер погоди. Сервер погоди приймає GET-запит і витягує необхідні дані, такі як параметри запиту. Він перевіряє правильність параметрів запиту і приводить їх до потрібного формату, якщо потрібно.

Сервер погоди робить запит до API погодного сервісу за допомогою отриманих параметрів. А API погодного сервісу обробляє запит та виконує операцію отримання погодних даних для вказаного місця розташування або міста. Далі API повертає відповідь з погодними даними на сервер. Сервер погоди приймає відповідь від API погодного сервісу та формує HTTP-відповідь для веб-додатку. Після чого сервер погоди надсилає HTTP-відповідь з погодними даними на веб-додаток у форматі JSON або XML.

Веб-додаток отримує відповідь від сервера погоди та обробляє отримані погодні дані відповідно до своєї логіки інтерфейсу користувача.

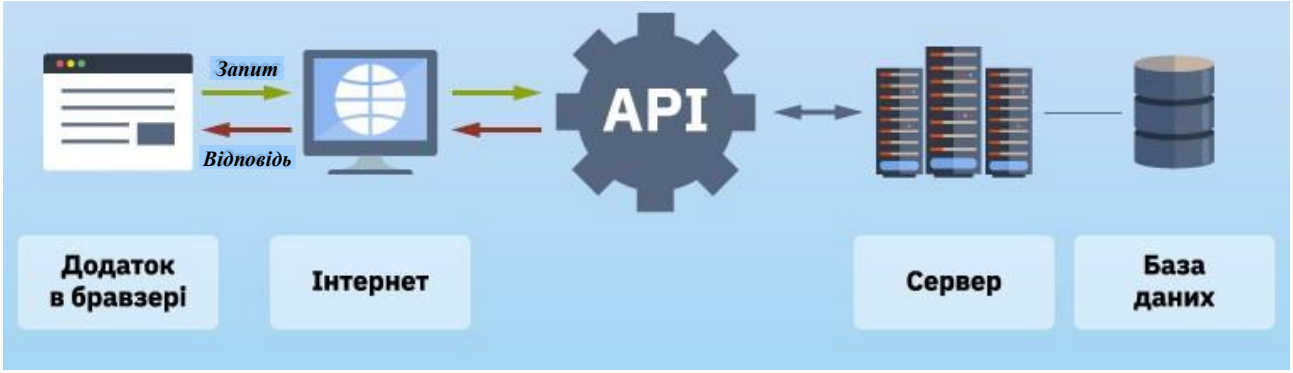


Рисунок 2.7 – Схема алгоритму GET-запиту веб-додатку до сервера з метеоданими

На рисунку 2.7 зображено схему алгоритму GET-запиту веб-додатку на сервер погоди та отримання відповідних даних з нього.

2.4. Сценарій використання API із глобальних метеосервісів

Розглянемо використання API сервісу прогнозу погоди з метою отримання відповідного відгуку сервера із даними, а тоді їх відображення у нашому веб-додатку. Метою реалізації сценарію є те щоб користувачі, які заходять на наш сайт, могли бачити поточний стан вітру та температури повітря.

У нас немає власної метеорологічної служби, тому доведеться зробити кілька запитів у сервіс прогнозу погоди, щоб отримати цю інформацію. Для цього наведемо приклад використання ***Aeris Weather API*** оскільки він характеризується надійністю і широким переліком показників які надаються сервером.

Отже для отримання коду запиту із Aeris Weather API, можна скористатися автоматичним генератором коду запиту – Request URL. Для цього слід виконати такі дії:;

- відкриваємо сайт – https://www.aerisweather.com/
- переходимо до розділу документації, клікнувши на назву розділу `Documentation`;
- натискаємо на `Weather API`;
- натискаємо на `Data Endpoints`;
- натискаємо на `Reference` в бічному меню і вибираємо `Endpoints`;
- в списку кінцевих точок оберемо `observations`;

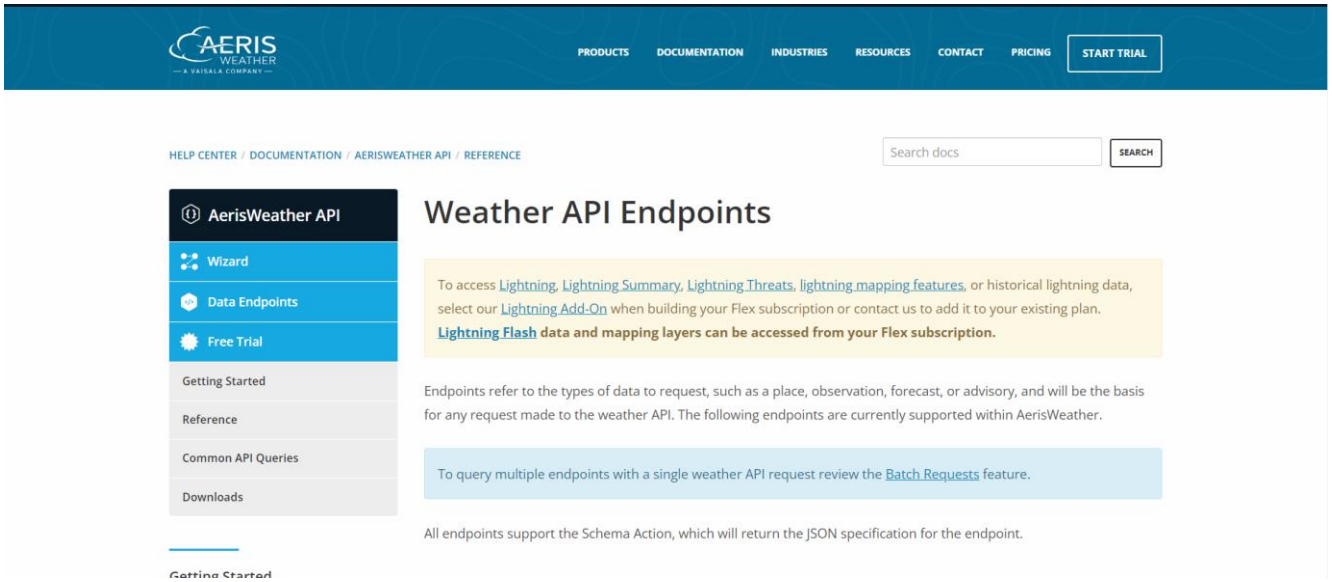
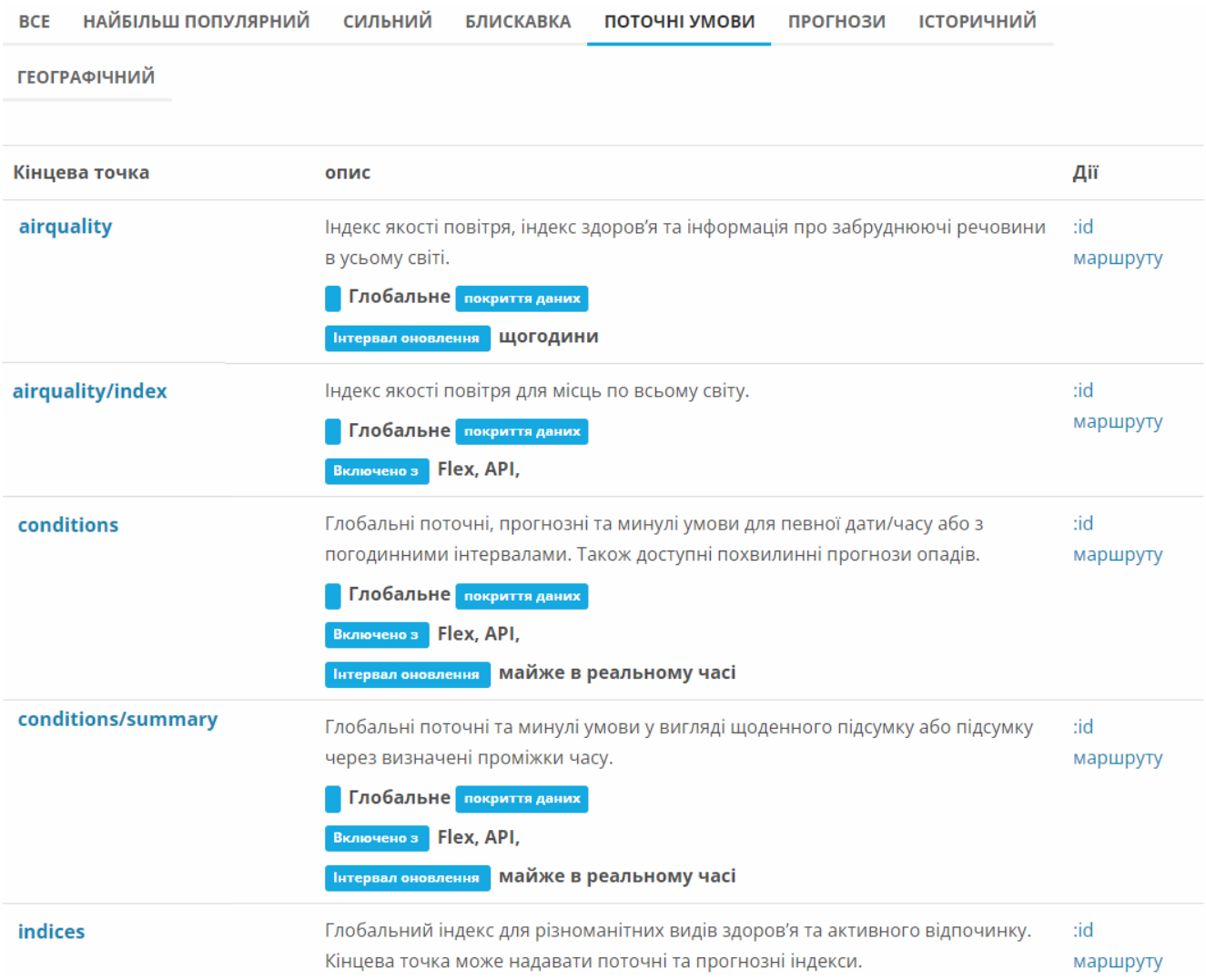


Рисунок 2.8 – Вікно вибору `Endpoints` у сервісі Aeris Weather API

Слід звернути увагу на тип інформації, яка доступна через кінцеву точку - `Endpoints`. Кінцеві точки `Endpoints` стосуються типів запитуваних даних, таких як місце, спостереження, прогноз або порада, і будуть основою для будь-якого запиту, зробленого до API погоди. AerisWeather підтримує дуже широкий набір `Endpoints`.



ВСЕ | НАЙБІЛЬШ ПОПУЛЯРНИЙ | СИЛЬНИЙ | БЛИСКАВКА | ПОТОЧНІ УМОВИ | ПРОГНОЗИ | ІСТОРИЧНИЙ | ГЕОГРАФІЧНИЙ

Кінцева точка	опис	Дії
airquality	Індекс якості повітря, індекс здоров'я та інформація про забруднюючі речовини в усьому світі. Глобальне покриття даних. Інтервал оновлення щогодини	:id маршруту
airquality/index	Індекс якості повітря для місць по всьому світу. Глобальне покриття даних. Включено з Flex, API,	:id маршруту
conditions	Глобальні поточні, прогнозні та минулі умови для певної дати/часу або з погодинними інтервалами. Також доступні похвилинні прогнози опадів. Глобальне покриття даних. Включено з Flex, API, Інтервал оновлення майже в реальному часі	:id маршруту
conditions/summary	Глобальні поточні та минулі умови у вигляді щоденного підсумку або підсумку через визначені проміжки часу. Глобальне покриття даних. Включено з Flex, API, Інтервал оновлення майже в реальному часі	:id маршруту
indices	Глобальний індекс для різноманітних видів здоров'я та активного відпочинку. Кінцева точка може надавати поточні та прогнозні індекси.	:id маршруту

….

Набір даних спостережень `observations` надає доступ до поточних і архівних даних спостережень за погодою з різноманітних станцій звітності. Перелік підтримуваних дій, звернень та параметрів API наведений в `Endpoint: observations` - https://www.aerisweather.com/support/docs/api/reference/endpoints/observations/#params

Основним джерелом даних спостережень є METAR, розташовані в

аеропортах або постійних метеостанціях. Звіти METAR генеруються раз на годину, але якщо умови істотно змінюються, то можуть видаватися додаткові спеціальні звіти. Інші джерела, такі як персональні метеостанції (PWS), можуть оновлюватися частіше, але не є офіційними станціями, які використовуються NOAA.

Отже, використовуючи API Wizard нами створено Request URL для нашого регіону, зокрема для міста Львів:

https://api.aerisapi.com/conditions/lviv,ua?format=json&plimit=1&filter=1min&client_id=[CLIENT_ID]&client_secret=[CLIENT_SECRET]

Цей URL надсилає запит API серверу Aeris Weather і отримуємо дані відповідь щодо погодних умов станом на 02.01.2024:


```
{
    "success": true,
    "error": null,
    "response": [
        {
            "loc": {
                "lat": 49.83826,
                "long": 24.02324
            },
            "place": {
                "name": "lviv",
                "state": "lv",
                "country": "ua"
            },
            "periods": [
                {
                    "timestamp": 1704192360,
                    "dateTimeISO": "2024-01-02T12:46:00+02:00",
                    "tempC": 4.23,
                    "tempF": 39.61,
                    "feelslikeC": 1.63,
                    "feelslikeF": 34.94,
                    "dewpointC": 1.22,
                    "dewpointF": 34.2,
                    "humidity": 81,
                    "pressureMB": 1010,
                    "pressureIN": 29.83,
                    "windDir": "W",
                    "windDirDEG": 263,
                    "windSpeedKTS": 10.56,
                    "windSpeedKPH": 19.55,
                    "windSpeedMPH": 12.15,
                    "windSpeedMPS": 5.43,
                    "windGustKTS": 23.35,
                    "windGustKPH": 43.24,
                    "windGustMPH": 26.87,
                    "windGustMPS": 12.01,
                    "precipMM": 0,
                    "precipIN": 0,
                    "precipRateMM": 0,
```

```
                        "precipRateIN": 0,
                        "snowCM": 0,
                        "snowIN": 0,
                        "snowRateCM": 0,
                        "snowRateIN": 0,
                        "snowDepthCM": 0,
                        "snowDepthIN": 0,
                        "pop": 0,
                        "visibilityKM": 12.2,
                        "visibilityMI": 7.581,
                        "sky": 96,
                        "cloudsCoded": "OV",
                        "weather": "Cloudy",
                        "weatherCoded": "::OV",
                        "weatherPrimary": "Cloudy",
                        "weatherPrimaryCoded": "::OV",
                        "icon": "cloudy.png",
                        "solradWM2": 84,
                        "uvi": 0,
                        "isDay": true,
                        "spressureMB": 975.4,
                        "spressureIN": 28.8,
                        "altimeterMB": 1008.6,
                        "altimeterIN": 29.79,
                        "solrad": {
                            "azimuthDEG": 184.3428,
                            "zenithDEG": 72.9232,
                            "ghiWM2": 84.1115,
                            "dniWM2": 7.7063,
                            "dhiWM2": 81.8485,
                            "version": "v2"
                        }
                    }
                ],
                "profile": {
                    "tz": "Europe/Kiev",
                    "tzname": "EET",
                    "tzoffset": 7200,
                    "isDST": false,
                    "elevM": 284,
                    "elevFT": 932
                }
            }
        ]
    }
```


Для надсилання запитів API та отримання відповідей від серверу Aeris Weather створено <head> код для нашого веб-додатку.

```
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="utf-8" />
        <title>Weather API</title>
        <meta   name="viewport"   content="initial-scale=1,maximum-scale=1,user-
scalable=no" />
        <script                                                              defer
src="https://cdn.aerisapi.com/sdk/js/latest/aerisweather.min.js"></script>
    </head>
    <body>
        <script>
```

```
    window.addEventListener('load', () => {
        const aeris = new AerisWeather('[CLIENT_ID]', '[CLIENT_SECRET]');
        const request =
            .format('json')
            .plimit(1)
            .filter('1min');
        request.get().then((result) => {
            console.log(result);
        });
    });
    </script>
</body>
</html>
```

Для нашого сценарію (показ прогнозу погоди на веб-сайті), ми могли б використовувати десятки різних API погоди. Загалом, існує велика кількість API інших прогнозних сервісів які ми також можемо використати для аналізу даних і представлення погодних умов для наших завдань. API-інтерфейси значно різняться за своїм дизайном, управлінням, відповідями та іншими деталями. Для більшого порівняння подивимося на наступні API погоди:

- API OpenWeatherMap – https://openweathermap.org/
- Dark Sky API – https://support.apple.com/en-us/102594
- Accuweather API – https://developer.accuweather.com/
- Weather Underground API – https://www.wunderground.com/weather/api/
- Weatherbit API – https://www.weatherbit.io/api

Кожен сервіс погоди має свій підхід до документування API. Як ми побачимо, різноманітність та унікальність кожного сайту, присвяченого API (навіть при наближенні до однієї й тієї ж теми – прогнозу погоди) створює багато проблем для команд технічних працівників. Змінюються не лише стилі веб-сайтів, а й термінологія API та словниковий запас для опису подібних концепцій.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ВІДОБРАЖЕННЯ ЗВЕДЕНИХ ДАНИХ ОНЛАЙН-СЕРВІСІВ СТЕЖЕННЯ ТА ПРОГНОЗУ ПОГОДИ

3.1. Вибір *Framework* та інтегровані рішення в інформаційній системі

Відповідно до завдань кваліфікаціної роботи нами описано методику отримання даних від API глобальних метеосервісів. Завдяки цьому, можна надавати інформацію щодо погодних умов для регіонів які нас цікавлять, а також узагальнювати інформацію з різних метеосервісів для порівняння даних та більш точної оцінки прогнозу погоди у нашому веб-додатку.

Опишемо розробку нашого веб-додатку, якиі надаватиме порівняльну інформацію про погоду зрізних джерел API.

Стандарти веб-розробки постійно зростають разом зі складністю сучасних технологій. За допомогою фреймворків складність розробки веб сервісів зменшується. В цьому розділі буде обрано фреймворк для розробки веб сервісу для відображення погоди.

Для аналізу було підібрані найбільш популярні фреймоврки для створення веб-сервісів:

- ✓ Vue;
- ✓ Angular;
- ✓ React;
- ✓ JQuery;
- ✓ Node;
- ✓ Titanium.

Під час рейтингового оцінювання фреймоврків були визнічені плюси та мінуси (табл. 3.1) на основі коментарях користувачів даного ПЗ [4, 22, 23, 26].

Vue – це веб-фреймворк призначений для розробки інтерфейсів на мові програмування JavaScript. Vue створений для поступового впровадження та росширення вже існуючих сервісів. Він вирішує різні завдання рівня уявлення (view), спрощує роботу з іншими бібліотеками та дозволяє створювати складні односторінкові додатки (SPA, Single-Page Applications).

Для роботи з фреймворком, вам вже потрібно знати HTML, CSS і звичайно

ж JavaScript хоча б на базовому рівні.

Таблиця 3.1 Таблиця оцінювання фрейморків [4, 22, 23, 26]

Фреймворк	**Плюси**	**Мінуси**
Vue	Легка інтеграція в проекти з використанням інших бібліотек; Створення SPA-додатків	Компонентний підхід; Система рендеринга надає менше можливостей у порівнянні з іншими
Angular	Підтримуються різні елементи MVC; Гнучкий; Пакети для розробки API	API Angular величезна, і потрібно розібратися з багатьма концепціями
React	Free and O pen Source; Швидкий розвиток; Підтримує віртуальну функціональність DOM	Алгоритм Virtual DOM неточний і повільний; Потрібне складне асинхронне програмування при спілкуванні з сервером
JQuery	Широко використовується завдяки швидкій обробці; У всіх браузерах поводиться однаково	Безліч функцій, що полегшують роботу з DOM, вже реалізовані нативно; Може бути нестабільним
Node	Можливість застосовувати одну мову на клієнті і сервері; Технологія стрімко поліпшується	Треба постійно стежити за оновленнями, деякі речі виходять недостатньо протестованими
Titanium	Простота навчання та реалізації; Високопродуктивна структура	Неефективний підхід до створення UI, на відміну від того ж React

Vue.js надає чудову можливість для реалізації сучасної архітектури MVC використовуючи шаблони та підключення компонентів до проекту. Принцип MVC у програмуванні (Model-View-Controller, Модель-Подання-Контролер) – одна з найбільш вдалих ідей. На перший погляд принцип MVC зрозумілий на інтуїтивному рівні, але не дуже простий якщо поглибитись в деталі (рис. 3.1) [4].

Такий підхід призводить до структурованого коду та дозволяє працювати над проектом більш спеціалізованим командам розробників, робить його більш зрозумілим і логічним, спрощує підтримку коду. Зміна в одному компоненті

мінімально впливає на інші. До однієї моделі можна підключати різні контролери та різні види.

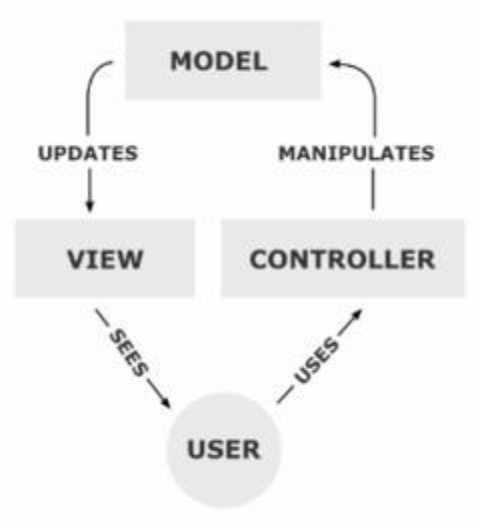


Рисунок 3.1 – Модель MVC

1. За допомогою Model ми інкапсулюємо дані додатка.
2. View відповідає за надання даних моделі та генерує DOM, який інтерпретується браузером клієнта.
3. Controller в цілому відповідає за обробку запитів і побудови відповідної моделі та передає його в представлення для рендеру.

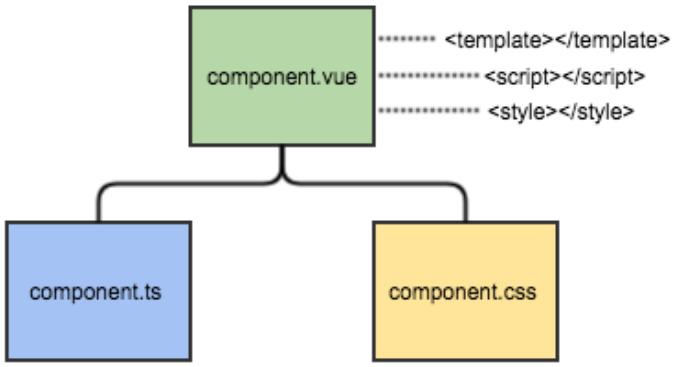


Рисунок 3.2 – Компонентна структура vue.js [26]

Vue дозволяє розділяти весь код програми на компоненти і збирати їх в єдиний додаток. Компоненти можна використовувати повторно в будь-яких інших додатках.

Компонент – це екземпляр Vue з попередньо встановленими опціями

показано на рис. 3.2 [4, 26].

Для підключення сторонніх бібліотек будем використовувати пакетний менеджер NPM (Node Package Manager) показаний на рис. 3.3 [4, 26].

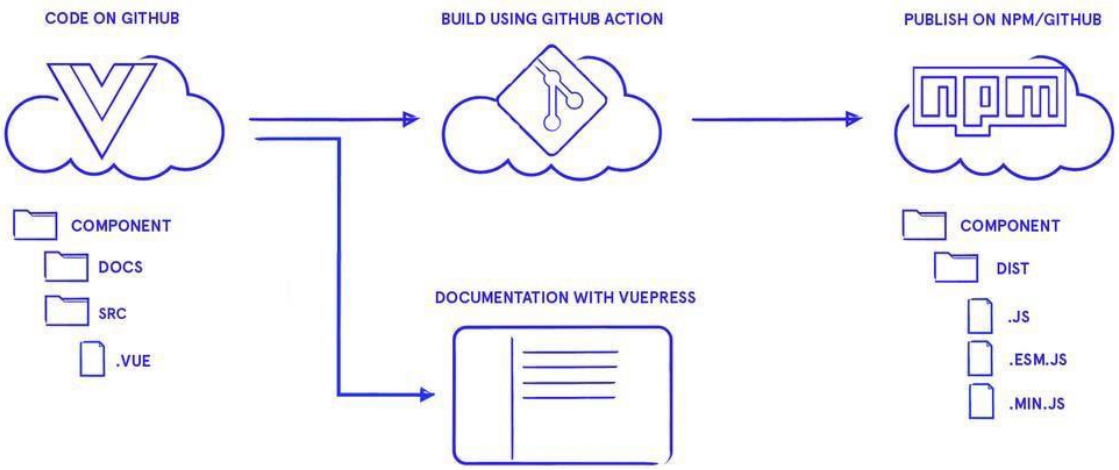


Рисунок 3.3 – NPM (Node Package Manager)

Поєднання вибраних методів рішення в одну інформаційну технологію.

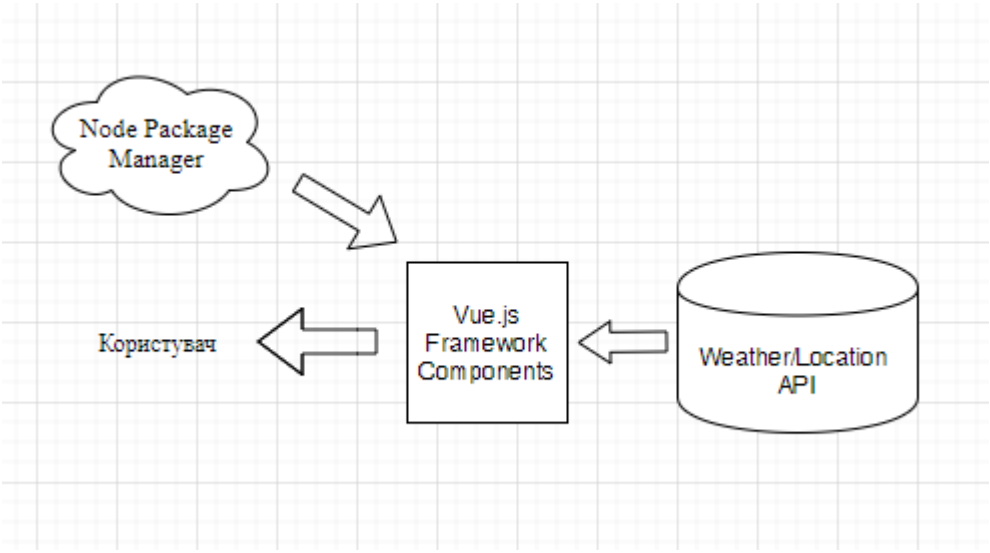


Рисунок 3.4 – Інформаційна технологія для метеосервісу

Проаналізувавши та вибравши компоненти для проектування метеосервісу було створено наступну технологічну схему роботи додатку, котра реалізовує ІС, відображено на рис. 3.4.

3.2. Розробка діаграми програмної реалізації інформаціної системи

Робота над додатком починається зі створення діаграми послідовності, діаграма варіантів використання, діаграми класів. Створення діаграми представлені в цьому підрозділі.

Діаграма послідовності (*англ.* **Sequence diagram**) – діаграма, на якій для деякого набору об'єктів на єдиній тимчасової осі показаний життєвий цикл будь-якого певного об'єкта (створення-діяльність-знищення якоїсь сутності) і взаємодія акторів (дійових осіб) ІС в рамках якого- або певного прецеденту (відправка запитів і відповідей) отримання [24, 25, 28].

Діаграма послідовнестой додатку для створення метеосервісу відображена на рис. 3.5.

Діаграма варіантів використання (*англ.* Use case diagram) в UML - діаграма, що відображає відносини між акторами і прецедентами і є складовою частиною моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

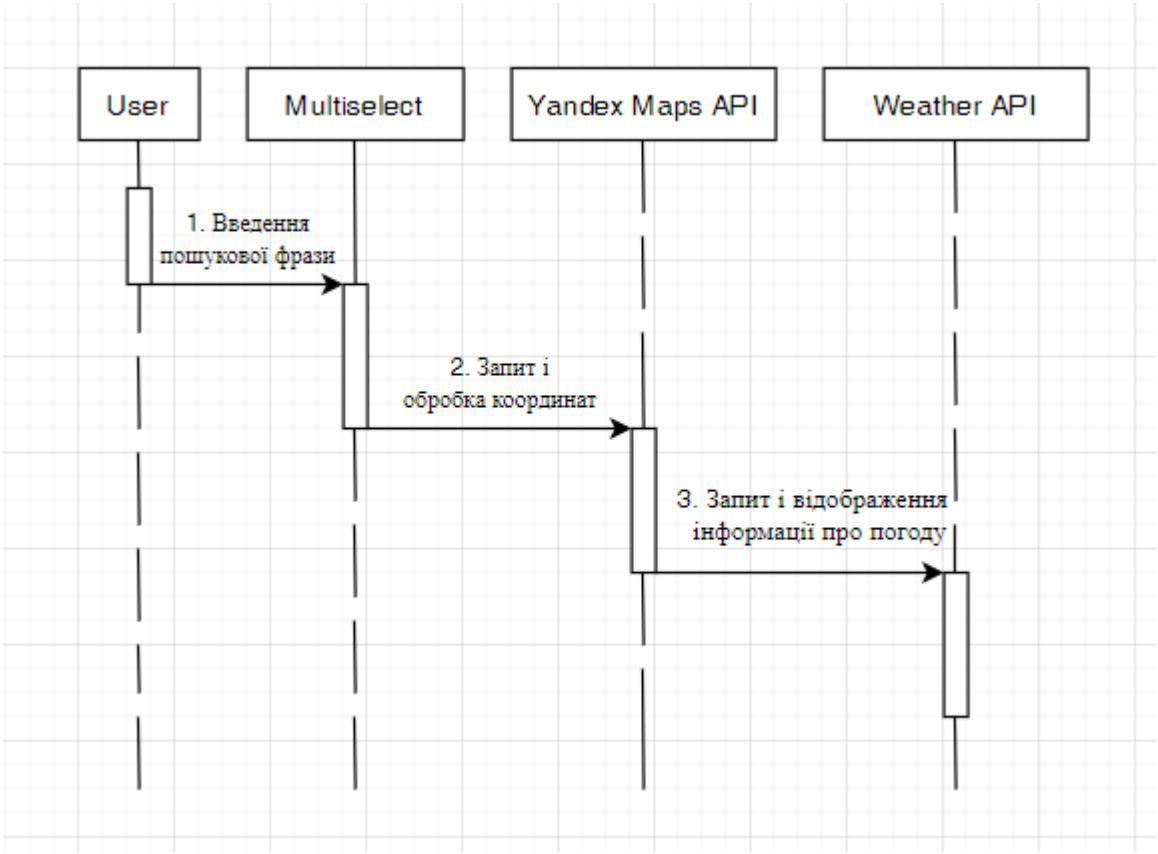


Рисунок 3.5 – Діаграма послідовнестой метеосервісу

Прецедент – можливість модельованої системи (частина її функціональності), завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат [24, 25, 28]. Прецедент відповідає окремому сервісу системи, визначає один з варіантів її використання і описує типовий спосіб взаємодії користувача з системою. Варіанти використання зазвичай застосовуються для специфікації зовнішніх вимог до системи.

Діаграми варіантів використання застосовуються при бізнес-аналізі для моделювання видів робіт, виконуваних організацією, і для моделювання функціональних вимог до ПС при її проектуванні і розробці. Побудова моделі вимог при необхідності доповнюється їх текстовим описом. При цьому ієрархічна організація вимог представляється за допомогою пакетів *use cases*.

На рис. 3.6 представлена спроектована діаграма для веб додатку.

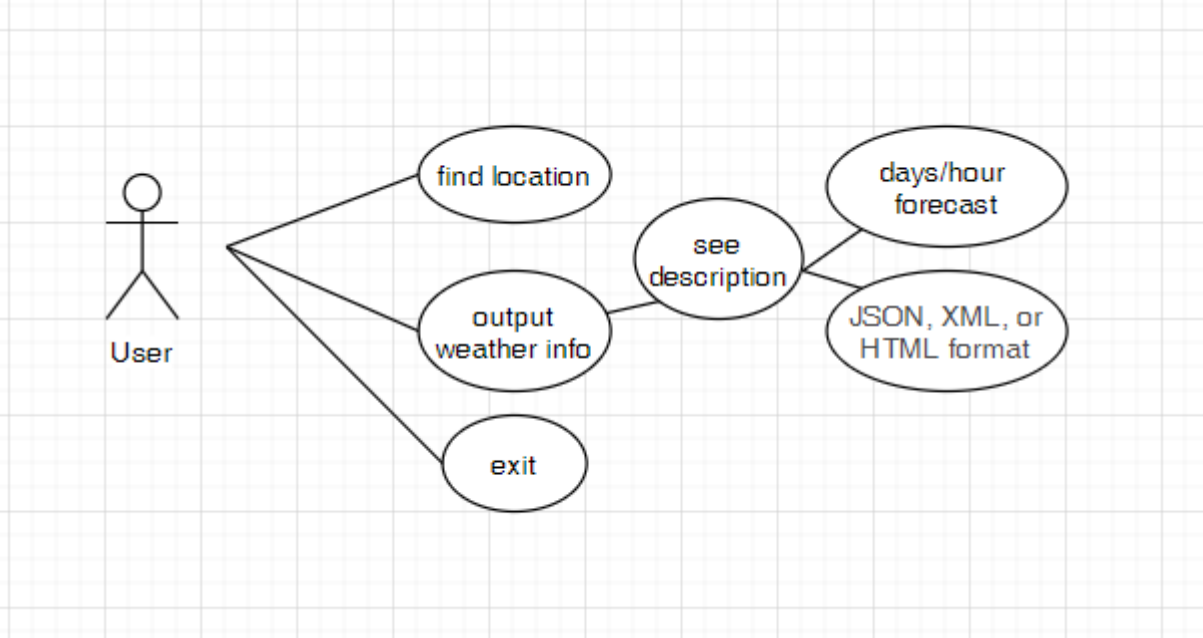


Рисунок 3.6 – Діаграми варіантів використання метеосервісу

ER-модель (від *англ.* Entity-relationship model, модель «сутність - зв'язок») – модель даних, що дозволяє описувати концептуальні схеми предметної області [24, 25, 28]. ER-модель використовується при високорівневої (концептуальному) проектуванні баз даних. З її допомогою можна виділити ключові сутності і позначити зв'язки, які можуть встановлюватися між цими сутностями.

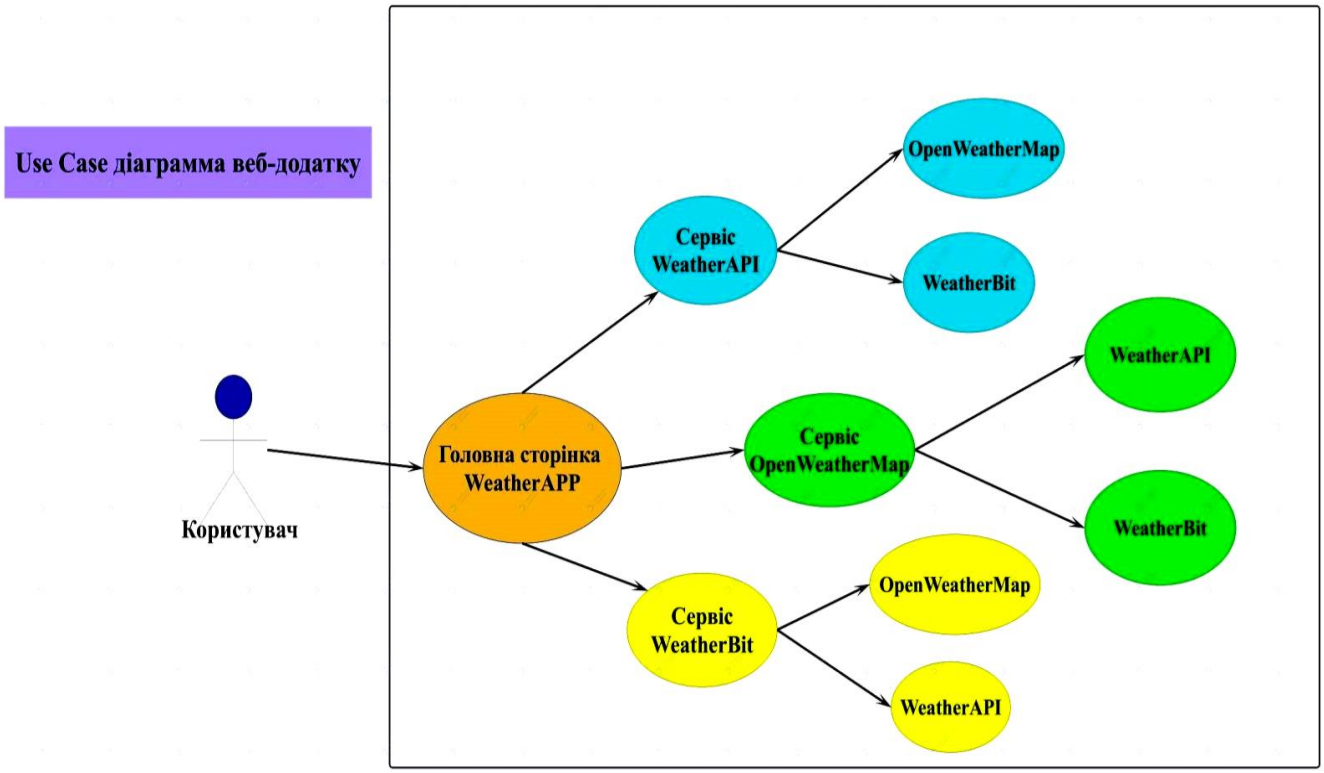


Рисунок 3.7 – Use Case діаграма веб-додатку

Під час проектування баз даних відбувається перетворення ER-моделі в конкретну схему бази даних на основі обраної моделі даних (реляційної, об'єктної, мережевий або ін.).

ER-модель являє собою формальну конструкцію, яка сама по собі не наказує ніяких графічних засобів її візуалізації. Як стандартна графічної нотації, за допомогою якої можна візуалізувати ER-модель, була запропонована діаграма "сутність-зв'язок» (*англ*. Entity-relationship diagram, ERD, ER-діаграма).

Поняття «ER-модель» і «ER-діаграма» часто вже не розрізняють, хоча для візуалізації ER-моделей можуть бути використані і інші графічні нотації, або візуалізація може взагалі не застосовуватися (наприклад, використовуватися текстовий опис).

Розроблена ER діаграма веб додатку представлена на рисунку 3.8. Створенні сутності Product – містить інформацію про продукт в музеї. Category – інформація про категроію продукту. Users – зберігає інформацію стосовно користувача. Authorities – якмими правами наділений користувач веб додатку.

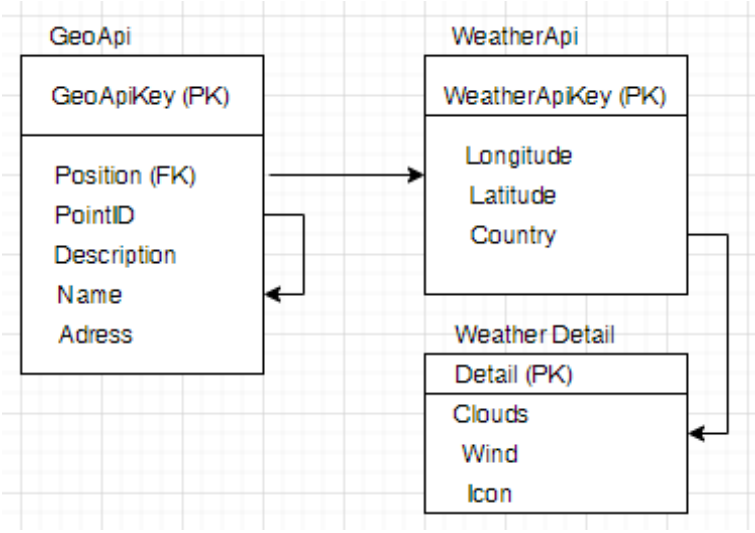


Рисунок 3.8 – ER-діаграма метеосервісу

Розробимо діаграму класів майбутнього веб додатку для реалізації метеосервісу.

Діаграма класів (*англ*. Static Structure diagram) – структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними [24, 25, 28]. Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого, або зворотного проектування.

Метою створення діаграми класів є графічне представлення статичної структури декларативних елементів системи (класів, типів і т.п.) Вона містить в собі також деякі елементи поведінки (наприклад – операції), проте їх динаміка повинна бути відображена на діаграмах інших видів (діаграмах комунікації, діаграмах станів). Для зручності сприйняття діаграму класів можна також доповнити поданням пакетів, включаючи вкладені.

При поданні сутностей реального світу розробнику потрібно відобразити їх поточний стан, їх поведінку і їх взаємні відносини. На кожному етапі здійснюється абстрагування від незначних деталей і концепцій, які не належать до реальності (продуктивність, інкапсуляція, видимість і т.п.). Класи можна розглядати з позиції різних рівнів. Як правило, їх виділяють три основних: аналітичний рівень, рівень проектування і рівень реалізації:

- на рівні аналізу клас містить у собі тільки начерк загальних контурів системи і працює як логічна концепція предметної області або програмного продукту.
- на рівні проектування клас відображає основні проектні рішення щодо розподілу інформації і планованої функціональності, об'єднуючи в собі відомості про стан та операції.
- на рівні реалізації клас допрацьовується до такого виду, в якому він максимально зручний для втілення в вибраному середовищі розробки; при цьому не забороняється опустити в ньому ті загальні властивості, які не застосовуються на обраною мовою програмування.

Можна скільки завгодно сперечатися, який фреймворк краще, але не можна не визнати очевидне – вони всі базуються на компонентах. У React, в Vue, в Angular ви займаєтеся тим, що ділите своє додаток на невеликі частини і працюєте з ними як з самостійними одиницями.

Концепція компонентного підходу у фронтенді відкриває неймовірні можливості для повторного використання написаного коду. Тільки уявіть, ви створюєте компонент (наприклад прелоадер) для одного проекту, а потім використовуєте його у всіх інших, без всякого переписування, або рефакторингу.

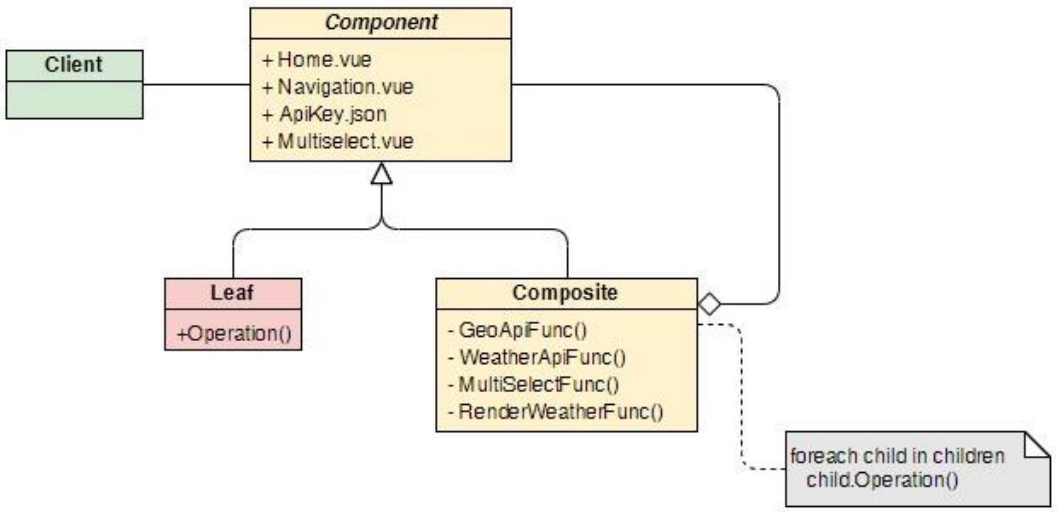


Рисунок 3.9 – Діаграма компонентів

Діаграма компонентів представлена на рис. 3.9. На ній можна виділити:

1. Компоненти, що описують моделі додатку;

2. Сервісний шар додатку;
3. Компоненти контролери;

За представлення відповідають Vue файли.

3.3. Розробка інформаційної системи у вигляді веб-орієнтованого додатку

На основі створених діаграм реалізуємо інформаційну систему (ІС) з елементами веб-орієнтованого додатку для оцінення функціонування метеосервісу. Структура програмної реалізації проекту метеосервісу зображена на рис. 3.10.

Проект метеосервісу складається з 5-ти основних папок:

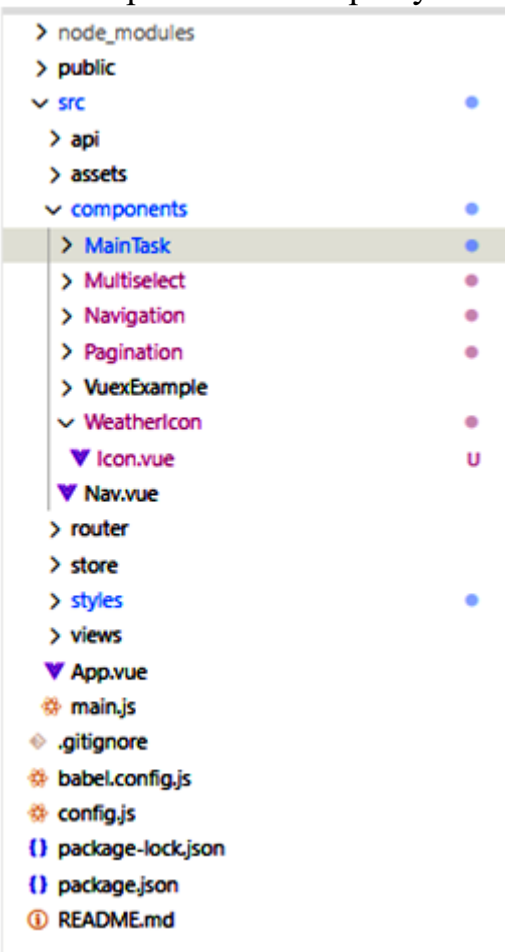


Рисунок 3.10 – Проект веб-орієнтованого додатку метеосервісу

node_modules – містить головні компоненти фреймоврку Vue;

components – містить компоненти та модулі нашого метеосервісу;

views – містить інформацію для відображення веб сторінок та файли конфігурації для запуску метеосервісу;

router – бібліотека маршрутизації;

store – контейнер станів додатку.

В сучасних проектах переважно використовують системи збірки. Для даного додатку буде задіяний NPM (Node Package Manager). Перед початком написання коду потрібно підключити необхідні бібліотеки. Нижче наведено необхідний набір бібліотек для проекту

Axios, Bootstrap, moment, vue-carousel, multiselect, vue-router, Vuex, EsLint.

```
{
  "name": "WeatherService",
  "version": "0.1.0", "private": true, "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": { "axios": "^0.19.0",
    "bootstrap": "^4.4.1",
    "bootstrap-vue": "^2.1.0",
    "core-js": "^3.4.3",
    "es6-promise": "^4.2.8",
    "moment": "^2.24.0",
    "semantic-ui-card": "^2.3.1",
    "vue": "^2.6.10",
    "vue-carousel": "^0.18.0",
    "vue-moment": "^4.1.0",
    "vue-multiselect": "^2.1.6",
    "vue-router": "^3.1.3",
    "vuex": "^3.1.2"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "^4.1.0",
    "@vue/cli-plugin-eslint": "^4.1.0",
    "@vue/cli-plugin-router": "^4.1.0", "@vue/cli-service":
    "^4.1.0",
    "babel-eslint": "^10.0.3",
    "eslint": "^5.16.0",
    "eslint-plugin-vue": "^5.0.0",
    "node-sass": "^4.12.0",
    "sass-loader": "^8.0.0",
    "vue-template-compiler": "^2.6.10"
  },
                  "eslintConfig": { "root": true,
    "env": { "node": true
    },
    "extends": [ "plugin:vue/essential", "eslint:recommended"
    ],
    "rules": {},
    "parserOptions": { "parser": "babel-eslint"
    }
  },
  "browserslist": [
    "> 1%",
    "last 2 versions"
  ]
}
```

Для того, щоб Axios міг обробляти HTTP запити, потрібно прописати відповідні конфігураційні налаштування в файлі router.js:

```
import Vue from 'vue'
```

```
import VueRouter from 'vue-router' import Home from
'../views/MainTask.vue'
import VuexExample from '@/views/AppVuex.vue' import Chapter1
from '@/views/Multiselect.vue' import Chapter2 from
'@/views/Navigation.vue' import Chapter3 from
'@/views/Pagination.vue'
import Navigation from '@/views/WeatherIcon.vue' // ---> тоді в
шляху замість import вставляю 'Navigation'

Vue.use(VueRouter) const routes = [
  {
    path: '/',
    name: 'WeatherServise',
    component: Home
  },
  {
    path: '/vuex',
    name: 'VuexExample',
    component: VuexExample
  },
  {
    path: '/Multiselect,
    name: Multiselect,
    component: Multiselect
  },
  {
    path: '/Pagination,
    name: Pagination,
    component: Pagination
  },
  {
    path: '/WeatherIcon, name: WeatherIcon,
    component: WeatherIcon
  },
  {
    path: '/Navigation',
    name: 'Navigation', component: Navigation
  },
  {
  path: '/weather',
  name: 'weather', component: Home
  }
]
const router = new VueRouter({ mode: 'history',
  base: process.env.BASE_URL, routes
})
export default router
```

В теці styles містяться конфігураційні файли для налаштування стилів нашого сервісу (рис. 3.11).

Після налаштування стилів переходимо до створення компонентів додатку. На рисунку 3.11 показана структура папки components. Можна виділити такі папки як Multiselect, Navigation, Pagination, Weather.

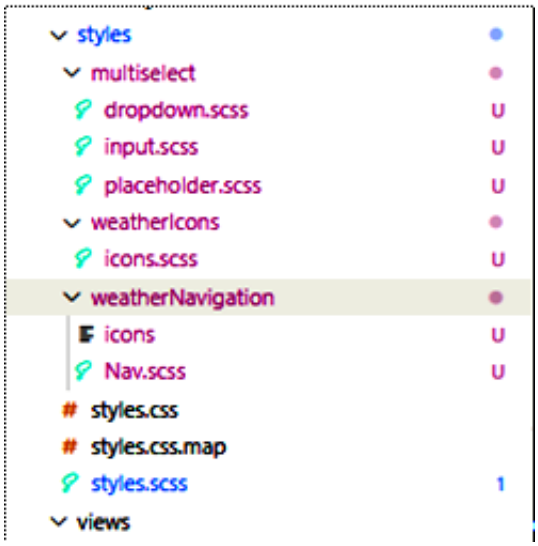


Рисунок 3.11 – Стилі веб-орієнтованого додатку ІС

Створюємо головний керуючий компонент на основі нашої ER-діаграми. Кожен з методів описує об'єкти у програмній реалізації ІС як веб-додатку.

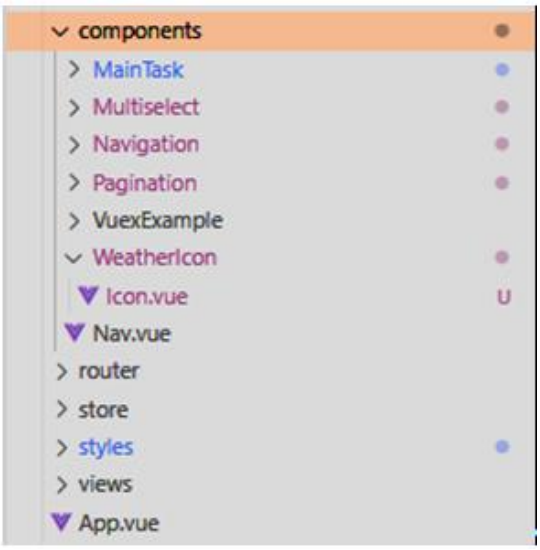


Рисунок 3.12 – Компонент веб-орієнтованого додатку

За допомогою методу displayWeather() відбуваєть рендер інтерфейсу звіту погоди:

```
displayWeather(value)
{ let sliceIndex = 0;
let sliceEnd;
   if (value && value.length) { this.errorWeatherData = false;
     this.weatherData = [];
     let currentDay = moment.unix(value[0].dt).utc().get('date');
// utc() - fix timezone
```

```
    value.forEach( (item, index) => {
      let itemDay = moment.unix(item.dt).utc().get('date');
      // utc() - fix
timezone
      sliceEnd = index;
      if (itemDay != currentDay)
      { this.weatherData.push(value.slice(sliceIndex, sliceEnd));
        currentDay = itemDay;
        sliceIndex = index;
      }
     });
     this.weatherData.push(value.slice(sliceIndex));
                          } else {
     this.errorWeatherData = true;
     this.weatherData = [];
   }
   this.selectedValue = this.selected.name this.$router.push({
   query: {
     name: this.selected.name, lat: this.lat, lon: this.lon,
     page: this.slide
   }})
   .catch(err => {console.log(err)})
}
```

Приклад головних методів, які звертаючись до гео-API, обмінюються параметрами та зберігають інформацію звіту погоди, яку користувач може побачити – описано в пункті 2.4 кваліфікаційної роботи.

РОЗДІЛ 4
ПРАКТИЧНЕ ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОНЛАЙН-ПЛАТФОРМИ АНАЛІЗУ МЕТЕОДАНИХ

4.1. Результати практичного використання інформаційної системи

Розроблено веб-додаток із використанням API глобальних метеосервісів, у якому є головна сторінка, де знаходиться посилання на різні сервіси прогнозу погоди. У розробці веб-додатку використано такі технології:

HTML – мова гіпертекстової розмітки документів для перегляду результату у браузері.

CSS – мова для опису зовнішнього вигляду веб-сторінки.

JavaScript – мова програмування, котру використовують для створення інтерактивних веб-сторінок.

У розробленому веб-додатку, є веб-інтерфейс – це головний файл. У ньому є посилання на три різні сервіси пронозу погоди. При наведені на будь-який сервіс, він буде виділятися (рис. 4.1).

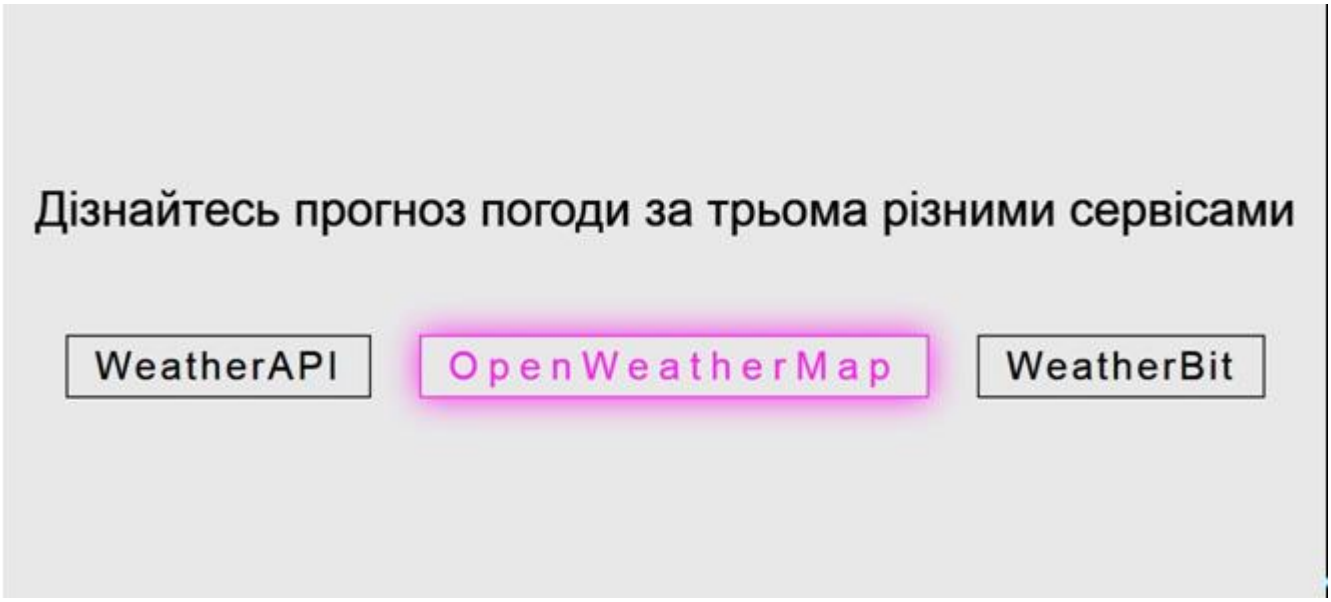


Рисунок 4.1 − Головна сторінка та веб-інтерфейс додатку

Після переходу на один с вище вказаних сервісів. Пишемо назву міста

бажано на англійській мові, бо може так статися що на сервісі, де береться прогноз погоди не буде іншої мови крім англійської і у відповідному полі натискаємо на Enter, або на кнопку (рис. 4.2)

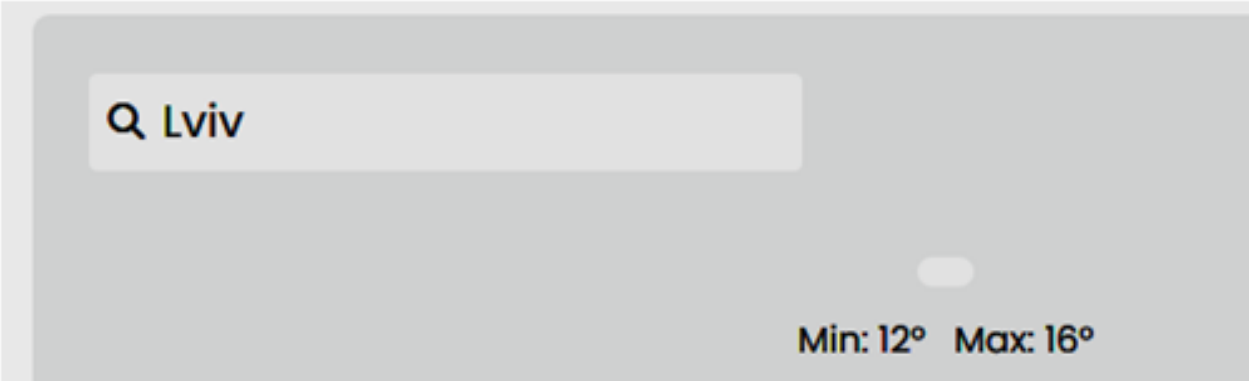


Рисунок 4.2 − Пошук міста

Якщо потрібно подивитися прогноз на п’ять днів, то потрібно на тиснути на кнопку “На Головну”, пишемо місто і натискаємо вже на кнопку “Forefast”. Отримуємо прогноз погоди на п’ять днів.

Прогноз погоди на п’ять днів (рис. 4.3).

Якщо потрібно подивитися або порівняти прогноз погоди на іншому сервісі. У самому верху є підписані дві кнопки, які ведуть на відповідний сервіс. Потрібно натиснути одну з них і відразу можна потрапити на інший сервіс (рис. 4.4).

Наприклад, сервіс WeatherAPI.

Рисунок 4.4 − Перехід на інший глобальний метеосервіс

Інтерфейс сервісу WeaptherAPI (рис. 4.5). Тут дещо інший дизайн так як це вже інший сервіс, але принцип пошуку і відображення залишився цей самий.

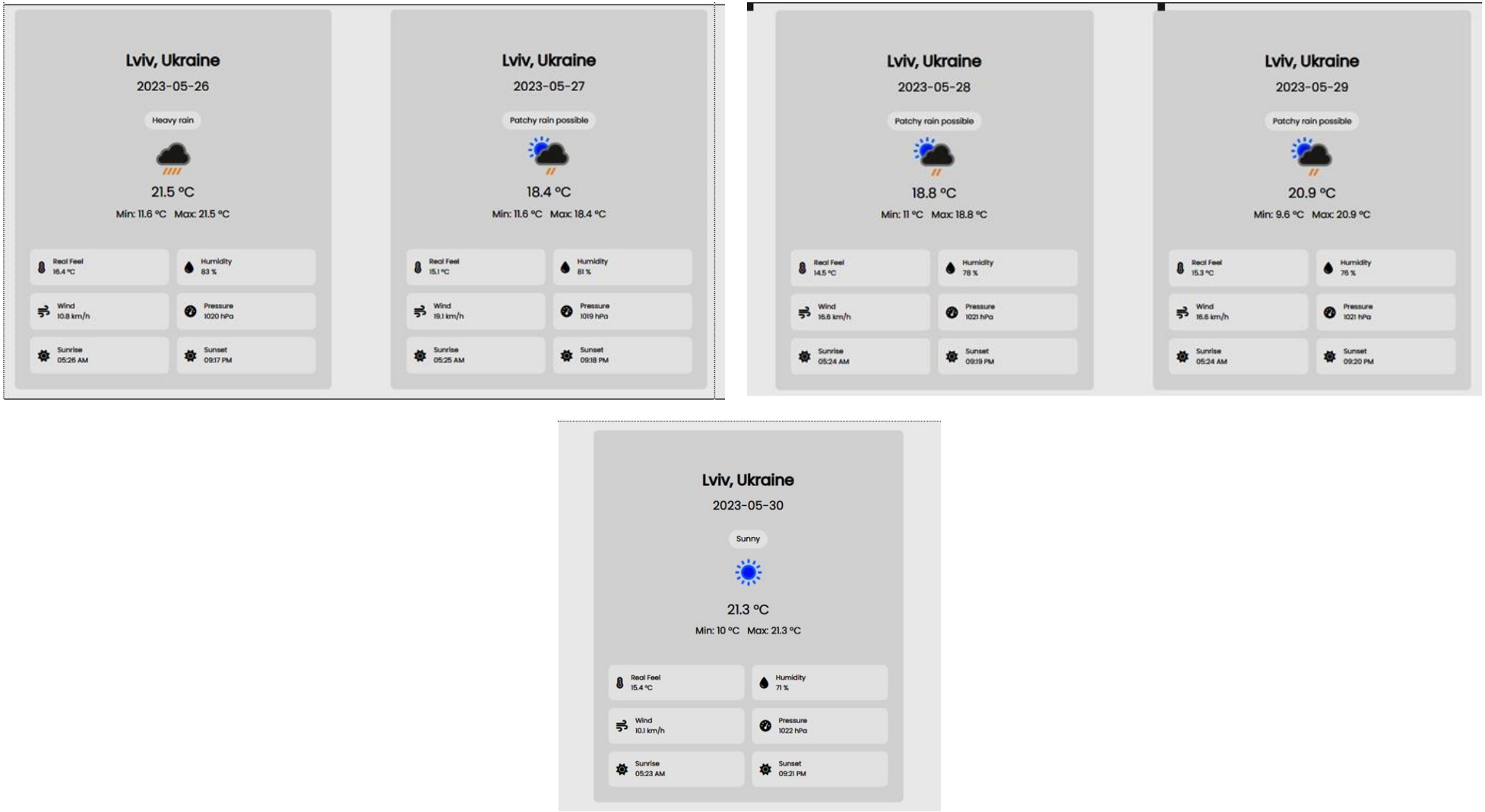


Рисунок 3.12 – Погода на п'ять днів

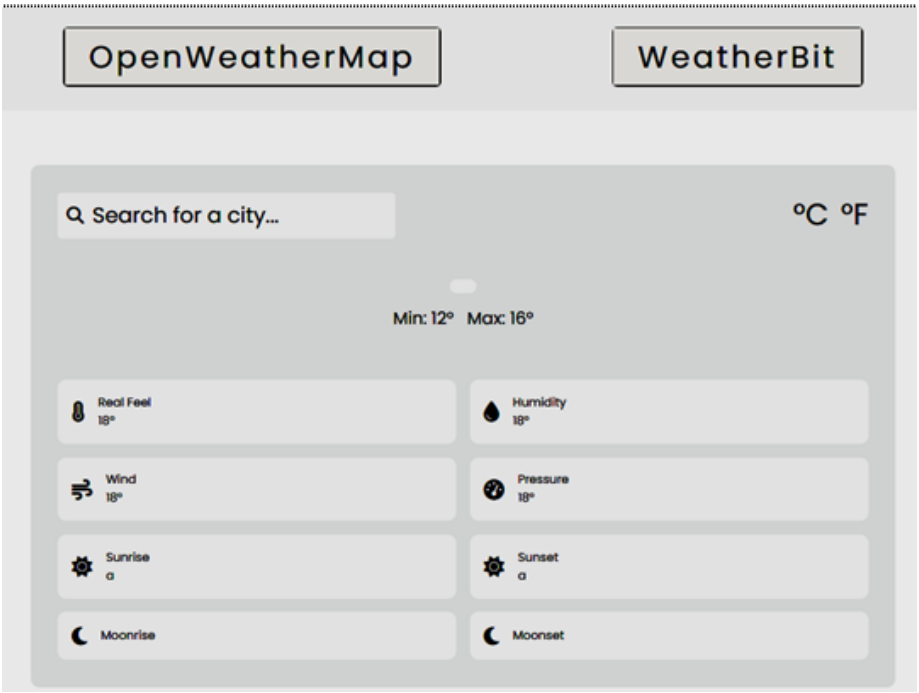


Рисунок 4.5 − Інтерфейс сервісу WeatherAPI

Дії аналогічні, як і в першому сервісі. В пошуковому полі вписуємо потрібне місто і натискаємо Enter, отримуємо прогноз погоди. Тут немає кнопок вибору на кількість днів, тут одразу показується прогноз погоди на один і на п’ять днів.

4.2. Порівняльна оцінка достовірності онлайн-платформ поточного аналізу метео-даних

Як уже зазначалося, метеодані можна отримати із локальних (приватних) метеостанцій (PWS), або з постійних метеостанцій які формують глобальні звіти METAR. Водночас, діючі глобальні метеосервіси вимагають легальної реєстрації тих чи інших метеостанцій, які використовуються NOAA. В США та Європеських країнах таких метеостанцій встановлено дуже велику кількість, в Україні ж порівняно значно менше, а на території Львівщини взагалі одиниці (рис. 4.).

Окрім того, кожен із глобальних метеосервісів використовує додатково інші джерела інформації, а також свої алгоритми прогнозування погоди.

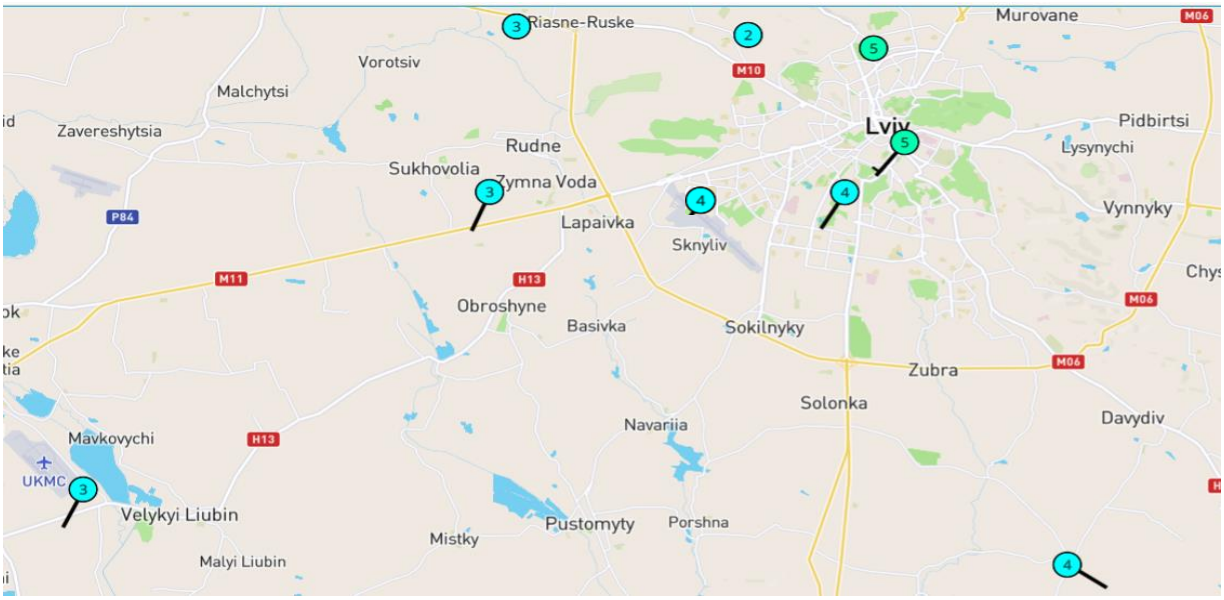


Рисунок 4.6 – Дев’ять метеостанцій реального часу Львівщини які формують метео-дані для глобальних сервісів (Wundermap)

Нами виконано порівняння даних чотирьох метеосервісів (рис. 4.7).

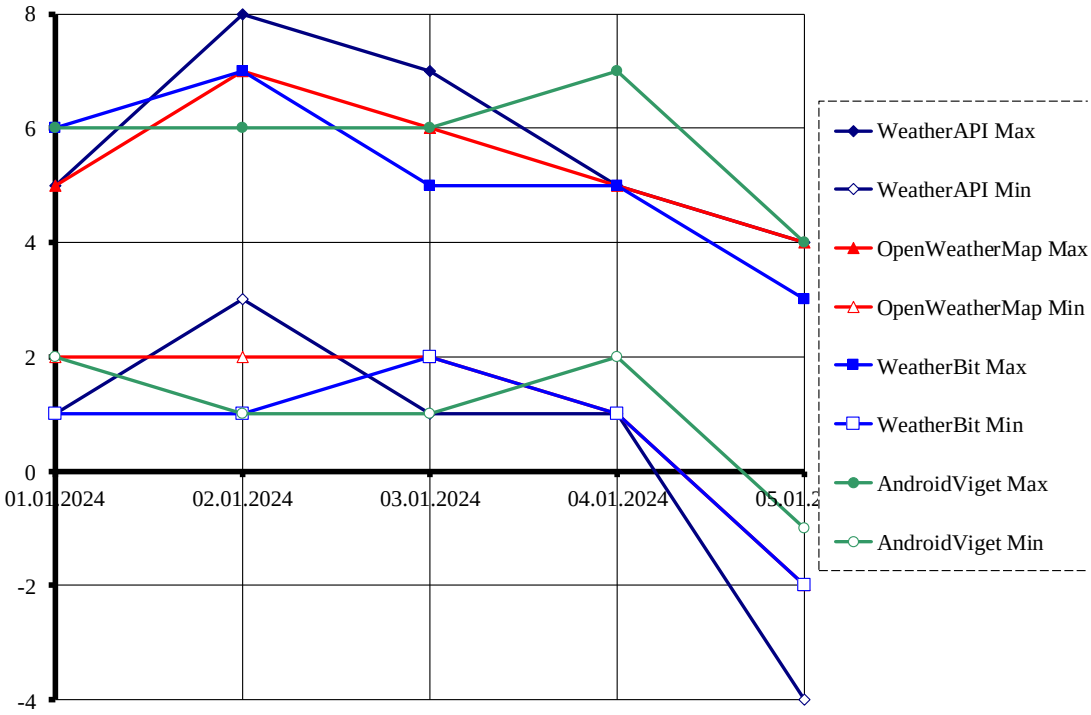


Рисунок 4.7. – Результати порівняння поточних (01.01.2024) і прогнозних даних метеосервісів для заданого календарного періоду

Як видно, усереднені показники температури відрізняються мінімально. Отримані результати дають підстави стверджувати, що розроблена нами інформаційна система моніторингу онлайн-платформ поточного аналізу метео-даних є досить коректною та відображає достовірні дані погоди.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта []. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній іх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: P_{13} = 0,2+ 0,4 – 0,2·0,4 = 0,0592.

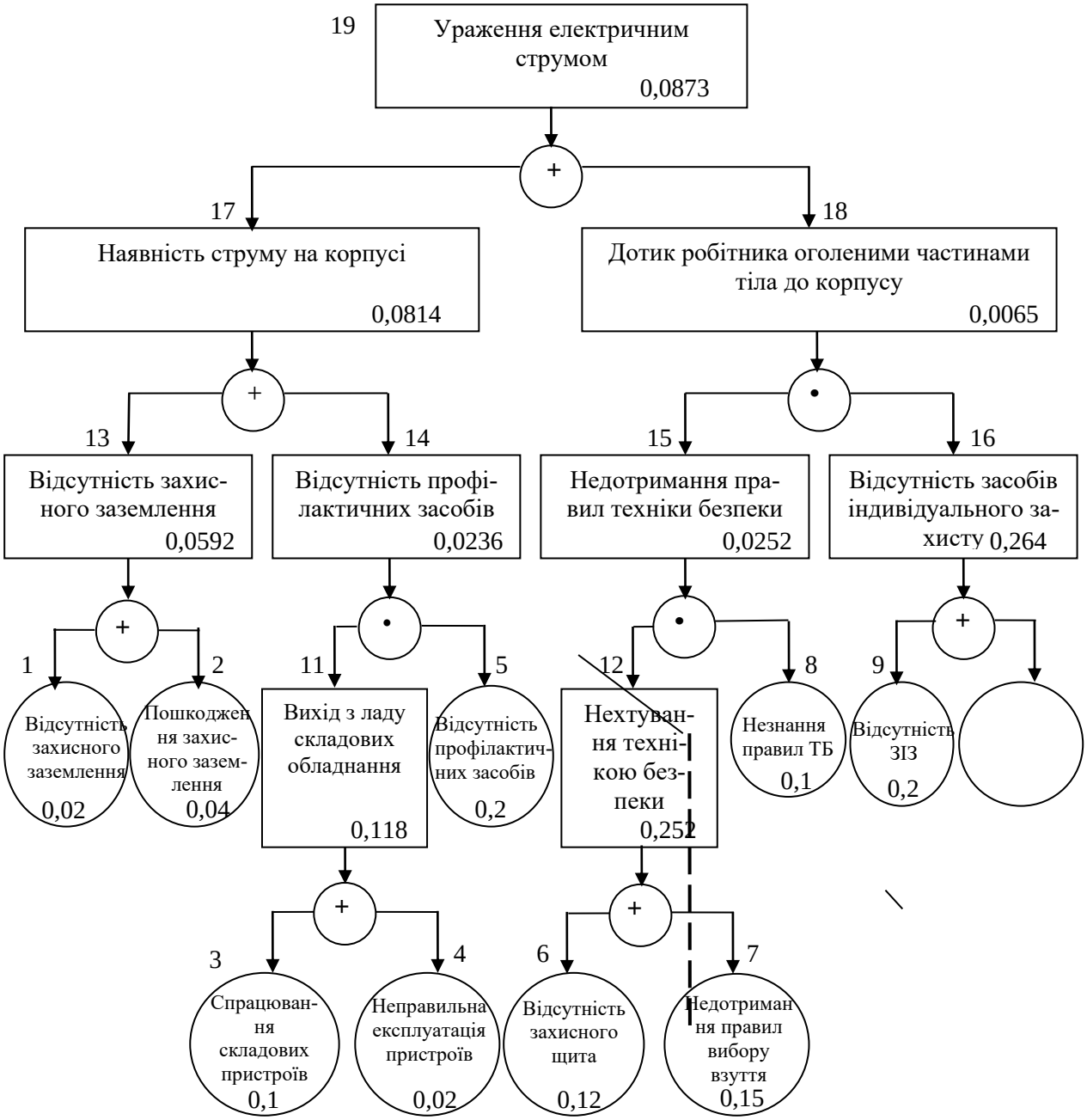


Рис. 5.1. Матриця логічних взаємозв’язків між окремими подіями травмонебезпечної ситуації []

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0{,}3 + 0{,}4 - 0{,}3 \cdot 0{,}4 = 0{,}118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0{,}3 + 0{,}5 - 0{,}3 \cdot 0{,}5 = 0{,}252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$

$$P_{17} = P_{13}+P_{14} - P_{13}\cdot P_{14} = 0{,}592+0{,}0236-0{,}0592\cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15}\cdot P_{16} = 0{,}264\cdot 0{,}0252 = 0{,}0065.$$

$$P_{19} = P_{17}+P_{18} - P_{17}\cdot P_{18} = 0{,}0065+0{,}0814 - 0{,}0065\cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Забезпечення захисту населення і території у разі загрози і виникнення надзвичайних ситуацій є одним з най важливіших завдань держави.

Захист населення є системою загально державних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами влад, органами управління з питань надзвичайних ситуацій та цивільного захисту населення, підпорядкованими їм системами, та підприємств, що забезпечують виконання організаційних, інженерно – технічних, санітарно – гігієнічних, проти епідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєво важливих інтересів громадян, держави, суспільства поділяють на зовнішні та внутрішні, виконують під час надзвичайних ситуацій техногенного та природного характеру та воєнних конфліктах.

Принципи захисту випливають з основних положень Женевської конвенції щодо захисту жертв війни та додаткових протоколів до неї, можливого характеру воєнних дій, реальних можливостей держави щодо створення матеріальної бази захисту. З метою захисту населення, зменшення втрат та шкоди економіці в разі виникнення надзвичайних ситуацій має право проводитись спеціальний комплекс заходів.

Оповіщення та інформування, яке досягається завчасним створенням і підтримкою в постійній готовності загально державної,територіальних та об’єктивних систем оповіщення населення.

ВИСНОВКИ

1. Метеосервіс (або веб-сайт погоди) – тип веб-сайту, який спеціалізується на представленні звітів про поточний стан погоди, а також виконує її прогноз. Всі матеріали, які можуть бути презентовані в подібному сервісі мають різний характер і вид. Метеосервіси з підтримкою інтернет-технологій можуть вирішувати класичні метеорологічні труднощі – зберігання, безпека, забезпечення широкого, швидкого і легкого доступу інформації.

2. Кожний популярний онлайн метео-сервіс надає послуги доступу до власного API та бази данних звітів погоди та геоположень. Ця послуга буває безкоштовна, але чаще всього щоб отримати доступ до API треба оформити платну підписку та отримати ліцензований ключ доступу. Розробник має можливість ввести цей ключ до параметрів зверення запиту та отримати повний список інформації погоди у різних форматах: XML; CSV; JSON.

3. Сьогодні є доступними глобальні онлайн-сервіси, які дозволяють стежити та прогнозувати погодні, метеорологічні, кліматичні умови та явища. Їх існує чимало і загалом дають змогу вирішувати різні задачі, зокрема, OSINT/GEOINT.

4. Принцип роботи клієнта у веб-додатку формує GET-запит, вказуючи необхідну URL-адресу сервера погоди, API ключ, а також місто або його координати. Після чого цей GET-запит відправляється на сервер погоди. Сервер погоди приймає GET-запит і витягує необхідні дані, такі як параметри запиту. Він перевіряє правильність параметрів запиту і приводить їх до потрібного формату, якщо потрібно.

5. Сервер погоди робить запит до API погодного сервісу за допомогою отриманих параметрів. А API погодного сервісу обробляє запит та виконує операцію отримання погодних даних для вказаного місця розташування або міста. Далі API повертає відповідь з погодними даними на сервер. Сервер погоди приймає відповідь від API погодного сервісу та формує HTTP-відповідь для веб-додатку. Після чого сервер погоди надсилає HTTP-відповідь з погодними даними

на веб-додаток у форматі JSON або XML.

6. Основним джерелом даних спостережень є METAR, розташовані в аеропортах або постійних метеостанціях. Звіти METAR генеруються раз на годину, але якщо умови істотно змінюються, то можуть видаватися додаткові спеціальні звіти. Інші джерела, такі як персональні метеостанції (PWS), можуть оновлюватися частіше, але не є офіційними станціями, які використовуються NOAA.

7. Для покращення обміну інструментами, установки різних модулів і управління їх залежностями обрано пакетний метод – Node Package Manager. Відповідно до цього, поєднання вибраних методів рішення в одну інформаційну технологію дало змогу створити технологічну можливість та справність інформаційної системи. Зазначену інформаціну технологію описано в розробленних схемах та діаграмах метеосервісу.

8. Програмна реалізація проекту ІС метеосервісу складається із 5-ти основних папок: 1) node_modules – містить головні компоненти фреймоврку Vue; 2) components – містить компоненти та модулі додатку; 3) views – містить інформацію для відображення веб сторінок та файли конфігурації для запуску веб додатку; 4) router – бібліотека маршрутизації; 5) store – контейнер станів. В сучасних проектах переважно використовують системи збірки. Для даного додатку буде задіяний NPM (Node Package Manager).

9. Відповіднодо до отриманих результатів, усереднені показники температури отримані з різнихметеосервісів відрізняються мінімально. Отримані результати дають підстави стверджувати, що розроблена нами інформаційна система моніторингу онлайн-платформ поточного аналізу метео-даних є досить коректною та відображає достовірні дані погоди.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Бекетов О.Г., Вітряк Є.А., Мироненко І.О., Овдій О.М. Розвиток інтернет-порталу метеорологічного прогнозування на мультипроцесорній платформі. Proceedings of the 10th International Conference of Programming UkrPROG'2016. 2016. 246-253.

2. Бідюк П.І. Моделювання та прогнозування нелінійних динамічних процесів / Бідюк П.І., Баклан І.В., Баклан Я.І., Коршевнюк Л.О. та ін. К.: ЕКМО, 2004. 120 с.

3. Болюбаш Ю.Я. Методи та засоби опрацювання великих даних у системах територіального управління / Ю. Я. Болюбаш // Науковий вісник Національного лісотехнічного університету: збірник наукових праць. – Львів : РВВ НЛТУ України. 2016. Вип. 26.4. С. 341-354.

4. Браун Є. Learning JavaScript: JavaScript E ssentials for Modern Application Development / Є. Браун, Коваленко В.А.(переклад), К.: Біном, 2017. 368 с.

5. Введення в пакетний менеджер NPM для початківців. (A Beginner's G uide). URL: http://prgssr.com/development/vvedenie-v-paketnyj -menedzher-npm-dlya- nachinayushih.html

6. Використовуємо Axios для доступу до API. URL: https://vuejs.org/v2/cook-book/using-axios-to-consume-apis.html

7. Дорошенко А.Ю., Бекетов О.Г., Прусов В.А., Тирчак Ю.М., Яценко О.А. Формалізоване проектування та генерація паралельної програми чисельного прогнозування погоди // Проблеми програмування. – 2014. – № 2–3. – С. 72–81.

8. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Павлючин Т.О., Вітряк Є.А. До створення Інтернет-порталу надання послуг метеорологічного прогнозування на мультипроцесорній платформі // Проблеми програмування. – 2015. – № 3. – С. 24–32.

9. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Яценко О.А. Автоматизоване проектування програм для розв'язання задачі

метеорологічного прогнозування. Проблеми програмування. 2016. № 1. С. 102–115.

10. Дорошенко А.Ю., Яценко О.А. Бекетов О.Г. Застосування графічних прискорювачів до задач метеорологічного прогнозування. // Теоретичні та прикладні аспекти побудови програмних систем. – 2014. – С. 101–105.

11. З нуля до деплоя: розробка системи документації з допомогою Vue і VuePress – https://medium.com/devschacht/vue-i-vuepress-cf6bde7c9a1f

12. Керівництво з Node.js, ч.1: загальні відомості і початокроботи. URL: https://habr.com/ua/company/uavds/blog/422893/

13. Ківганов А.Ф., Хоменко Г.В., Хохлов В.М., Бондаренко В.М. Гідродинамічні методи прогнозу погоди і сіткові методи їх реалізації. – Одеса: Одеський державний екологічний університет. 2002. 179 с.

14. Клімат України [За редакцією В.М. Ліпінського, В.А. Дячука, В.М. Бабіченко]. Київ: видавництво Раєвського, 2003. 344 с.

15. Крокфорд. Д. Як влаштований JavaScript: Навчальний пос. / Д. Крокфорд, К .: Міннесота, 2019. 304 с.

16. Лехман С.Д. та ін. Запобігання аварійності і травматизму у сільському господарстві / С.Д. Лехман, В.І. Рубльов, Б.І. Рябцев. К.: Урожай, 1993. 272 с.

17. Лихочвор В.В. Рослинництво. Технології вирощування сільськогосподарських культур. Львів: НВФ “Українські технології”, 2002. 800 с.

18. Метеосервіс «Gismeteo». URL: https://www.gismeteo.ua/ua/weather-lviv-4949/

19. Метеосервіс «Sinoptik». URL: https://ua.sinoptik.ua

20. Прусов В.А., Дорошенко А.Ю. Моделювання природних і техногенних процесів в атмосфері. – К.: Наукова думка, 2006. – 542 с.

21. Прусов В.А., Сніжко С. І. Математичне моделювання атмосферних процесів. – Київ: Ніка-Центр, – 2005. – 496 с.

22. Резіг Д. Секрети JavaScript / Резіг Д., Марас І., Бібо Б. К.: Вільямс,

2017. 544 с.

23. Робота з даними на межі Vue.js-додатку. Постановка задач – https://habr.com/ua/company/uavds/blog/505756/

24. Спірін О. М. Зміст навчального матеріалу спецкурсу "Хмарні інформаційно-аналітичні технології у науково-дослідному процесі". Інформаційні технології і засоби навчання. 2016. Т. 52, вип. 2. С. 108-120.

25. Спрощуємо роботу з npm: корисні скорочення та трюки для розробки. URL: https://tproger.ua/translations/npm-tricks/

26. Стандартні директиви в Vue.js. URL: https://monsterlessons.com/project/lessons/standartnye-direktivy-v-vuejs

27. ТОП сервісів спостереження за погодними умовами. URL: https://kr-labs.com.ua/blog/top-weather-forecast-services

28. Шаховська Н. Б. Організація великих даних у розподіленому середовищі / Н. Б. Шаховська, Ю. Я. Болюбаш, О. М. Верес // Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. 2014. № 2. С. 147-155.

29. Apache Hadoop: сайт. URL: http://hadoop.apache.org/ (дата звернення: 01.12.2023).

30. Apache Oozie Workflow Scheduler for Hadoop: сайт. URL: http://oozie.apache.org/ (дата звернення: 01.12.2023).

31. Composition API в Vue 3 – плюси, мінуси і досвід використання – https://tproger.ua/video/composition-api-in-vue/?autoplay=1

32. EOS Data Analytics: Проблеми на Землі – рішення в космосі. URL: https://eos.com/eos-crop-monitoring/

33. Giovannettone J.P. and Barros A.P. Probing Regional Orographic Controls of Precipitation and Cloudiness in the Central Andes Using Satellite Data. Journal of Hydrometeorology February. 2019. Vol. 10, N 1. P. 167-182.

34. MeteorJS : JavaScript APPS. – Mode of access : URL : https://www.meteor.com

35. Model-View-Controller. URL: https://wikipedia.org/wiki/Model-View-

Controller

36. Shpyg V. et al. The application of regional NWP models to operational weather forecasting in Ukraine. CAS Technical Conference (TECO) on “Responding to the Environmental Stressors of the 21st Century”: Conf. Materials. 2013. URL: http://www.wmo.int/pages/prog/arep/cas/documents/Ukraine-NWPModels.pdf (дата звернення: 27.11.2023).

37. SmartFarming – комплексний інтегратор технологій у рослинництві. URL: https://www.smartfarming.ua/

38. The R Project for Statistical Computing: сайт. URL: https://www.r-project.org/ (дата звернення: 01.12.2023).

39. Vue: як використати компоненти. URL: https://medium.com/@mo-dex13/vue-js-2-8f029ba5a60c

40. Warren S.G., Eastman R.M., Hahn C.J. A survey of Changes in Cloud Cover and Cloud Types over Land from Surface Observations, 1971-96. Climate. 2017. N 20. P. 717–738.

41. Wetter und Klima – Deutscher Wetterdienst – Startseite (веб-сайт метеорологічного центру у м. Офенбах) [Електронний ресурс]. – Режим доступу: http://www.dwd. de. – 25.02.2016 р.

ДОДАТКИ

Додаток А.
Фрагмент коду головної функції

main.js

```
import Vue from 'vue'
import axios from 'axios'
Vue.prototype.$axios = axios
import VueCarousel from 'vue-carousel';
Vue.use(VueCarousel);
import BootstrapVue from 'bootstrap-vue' Vue.use(BootstrapVue);
import moment from 'moment' Vue.prototype.moment = moment moment.locale('ua');
import App from './App.vue' import router from './router'
import store from './store' Vue.config.productionTip = false export const eventBus = new Vue()
new Vue({
    router, store,
    render: h => h(App)
}).$mount('#app')
```

App.vue

```
<template>
    <div id="app">
        <keep-alive><!—Кешуємо компоненти-->
        <router-view/>
        </keep-alive>
    </div>
</template>
<style lang="scss">
@import"./styles/styles.css";
#app {
    width: 100vw; height: 100vh;
    font-family: tahoma; color:#282828; margin: 0px;
}
body {
    margin: 0px;
}
</style>
```

Router.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/MainTask.vue'
import VuexExample from '@/views/AppVuex.vue' import Chapter1 from '@/views/Multiselect.vue' import Chapter2 from
'@/views/Navigation.vue' import Chapter3 from '@/views/Pagination.vue'
import Navigation from '@/views/WeatherIcon.vue' // ---> тоді в адресаті замість import ставлю 'Navigation'
Vue.use(VueRouter) const routes = [
{
      path: '/',
      name: 'WeatherServise', component: Home
},
{
      path: '/vuex',
      name: 'VuexExample', component: VuexExample
},
{
      path: '/Multiselect,
      name: Multiselect, component: Multiselect
},
{
      path: '/Pagination,
      name: Pagination, component: Pagination
```

```
},
{
        path: '/WeatherIcon,
        name: WeatherIcon, component: WeatherIcon
},
{
        path: '/Navigation',
        name: 'Navigation', component: Navigation
},
{
        path: '/weather',
        name: 'weather', component: Home
}]
const router = new VueRouter({ mode: 'history',
    base: process.env.BASE_URL,
    routes
})
export default router
            store.j s
import Vue from 'vue' import Vuex from 'vuex'
import cart from './modules/cart'
import products from './modules/products'
import createLogger from '../../../src/plugins/logger' Vue.use(Vuex)
const debug = process.env.NODE_ENV !== 'production'
export default new Vuex.Store({ modules: {
        cart, products
    },
    strict: debug,
    plugins: debug ? [createLogger()] : []
})
            weatherStore.j s
import shop from '../../api/weather'
const state = { items: [],
    checkoutStatus: null
}
// getters
const getters = {
    searchStates: (state, getters, rootState) => { return state.items.map(({ id, quantity }) => {
            const product = rootState.products.all.find(product => state.id === id)
            return {
                title: search.title,
                coordinate: search.coordinate
            }
        })
    },
    totalAim: (state, getters) => {
        return getters.totalAim.reduce((total, search) => {
            return total + search.name * search.adress
        }, 0)
    }
}
// actions
const actions = {
    checkout ({ commit, state }, search) {
        const savedAdress = [...state.items] commit('setCheckoutStatus', null) commit('setLon', { items: [] }) weather.api(
            search,
            () => commit('setCheckoutStatus', 'successful'), () => {
                commit('setCheckoutStatus','failed')
                // rollback to the api saved before sending the request commit('setLonAndLat', { items: savedSearch })
```

```
            }
        )
    },
    sendCoordinate ({ state, commit }, coordinate) { commit('setCheckoutStatus',null)
        if (item.inventory > 0) {
            const container = state.items.find(item => item.id === search.id)
            if (!coordinate) { commit('pushCoordinate',
               { id: coordinate.id })
            } else {
                commit('reset', item)
            }
            // remove 1 item from stock
            commit('products/'pushCoordinate, { id: item.id }, { root: true })
        }
    }
}
// mutations
const mutations = {
    pushCoordinate (state, { id }) {
        state.items.push({ id,
        })
    },
    resetCoordinate (state, { id }) {
        const coordinate = state.items.find(item => item.id === id)
        coordinate.item = []
    },
    setCartCoord (state, { items }) { state.items = items
    },
    setCheckoutStatus (state, status) { state.checkoutStatus = status
    },
    resetAll (state, status) { state = ...state
    }
}
export default { namespaced: true,
    state,
    getters,
    actions, mutations
}
```

Фрагмент коду ІС метеосервісу - WeatherMain.vue

```
<template>
        <div class="weather">
                <h1>ПОГОДА</h1>
        <multiselect
                placeholder="Введіть місто"
                selectLabel="Нажміть Enter, щоб выбрати"
                v-model="selected"
                :value='selectedValue'
                @select="getWeatherApi"
                @keyup="totalcharacter++"
                @search-change="getGeoApi"
                :options="options" label="name"
                track-by="name">
                <template slot="noResult">
                        <span>Введіть мінімум три символа, або спопробуйте змінити запит </span>
                </template>
                <template slot="noOptions">
                <span> Введіть мінімум три символа, або спопробуйте змінити запит </span>
                </template>
                </multiselect>
                <b-carousel v-if="weatherData.length" id="carousel-1"
                ref="myCarousel" v-model="slide"
                :interval="0" controls indicators @sliding-end="onSlideEnd">
                <b-carousel-slide v-for="weatherDay in weatherData":key="weatherDay.id">
                    <h1 class="day">{{ moment(weatherDay[0].dt_txt).format('dddd') }} <br>
 {{moment(weatherDay[0].dt_txt).format('LL') }}</h1>
<div class="example-slide">
                                <div v-for="weatherHour in weatherDay" :key="weatherHour.id"
 class="slide-item">
 <div class="weather-time">{{
 moment.unix(weatherHour.dt).utc().format('LT')}}</div>
                                        <!--{{weatherHour.weather[0].main}}-->
                                        <Icon :iconType="weatherHour.weather[0].main" />
                                        <div class="weather-temp">{{
 Math.round(weatherHour.main.temp) + ' °' + 'C' }}</div>
                                </div>
                        </div>
</b-carousel-slide>
        </b-carousel>
        </div>
 </template>
 <script>
 /* eslint-disable no-console */
 import Multiselect from 'vue-multiselect'
 import moment from 'moment'
 import config from '../../../config.js'
 import Icon from './Icon' export default {
        name: 'weather',
        components: { Multiselect, Icon }, data () {
   return {
                        totalcharacter : 0,
                        selected: {},
                        options: [],
                        geo: {},
                        weather: {},
                        dateTime: {},
                        coord: [],
```

```
                temp: [],
                time: [],
                weatherData: [], itemPerPage: 0, days: [],
                nextLabel: "<div class='nav-carousel-next'></div>", prevLabel: "<div class='nav-carousel-prev'></div>",
                lat: 0,
                lon: 0,
                page: 0,
                saveUrl: [], selectedValue: '', currentPage: 0,
                ccc: 0,
                slide: 0, sliding: null
            }
        },
        methods: {
            onSlideEnd() { this.sliding = false
                console.log(this.slide);
                this.$router.push({ query: { name: this.selected.name, lat:
this.lat, lon: this.lon, page: this.slide } }).catch(err => {console.log(err)})
            },
            getGeoApi(currentValue){
            this.temp = [];
            this.time = [];
            if(currentValue.length <= 2) {
            this.options = []; this.temp = [];
            }else{
            this.$axios.get(`${config.geoApi}?apikey=${config.apiKeyGeo}&format=json&geocode
            =${currentValue}`)
            .then( response => {
            console.log(`${config.geoApi}?apikey=${config.apiKeyGeo}&format=json&geocode=${c urrentValue}`);
            this.geo=response.data.response.GeoObjectCollection.featureMember; this.geo.forEach(obj => {
            this.options.push(obj.GeoObject);  //  витягнути всі обєкти з масиву api
            this.coord.push(obj.GeoObject.Point.pos); // координати
                });
                console.log(this.options)
                })
            }
            },
            getWeatherApi(currentValue){
                var coord = currentValue.Point.pos.split(" "); this.lon = coord[0];
                this.lat = coord[1]; this.axiosWeatherApi();
                },
            axiosWeatherApi() {
this.$axios.get(`${config.weatherApi}lat=${this.lat}&lon=${this.lon}&units=metri c&appid=${config.apiKeyWeather}`)
                .then( response => {
console.log(`${config.weatherApi}lat=${this.lat}&lon=${this.lon}&units=metric&ap pid=${config.apiKeyWeather}`);
                this.weather = response.data.list; this.weather.forEach(obj => {
             this.temp.push(obj.main.temp);  // температура в Кельвінах за 5 днів (кожні 3 год)
             this.time.push(obj.dt_txt);  //  дата  і  ч а с  (5  днів кожні 3 год - 40 елементів по 8 елементів на цілий
день)
                });
                this.displayWeather(this.weather);
                })
            },
            displayWeather(value) {
                let sliceIndex = 0;
                let sliceEnd;
                if (value && value.length) {
                    this.errorWeatherData = false; this.weatherData = [];
                    let currentDay = moment.unix(value[0].dt).utc().get('date'); //
utc() - fix timezone
```

```
                                value.forEach( (item, index) => {
                                        let itemDay = moment.unix(item.dt).utc().get('date'); //
  utc() - fix timezone
  sliceEnd));
//console.log(index); sliceEnd = index;
if (itemDay != currentDay) { this.weatherData.push(value.slice(sliceIndex,
        currentDay = itemDay; sliceIndex = index;
                                        }
                                        });
                                this.weatherData.push(value.slice(sliceIndex));
                        } else {
                                this.errorWeatherData = true; this.weatherData = [];
                        }
                        this.selectedValue = this.selected.name
                        this.$router.push({ query: { name: this.selected.name, lat:
  this.lat, lon: this.lon, page: this.slide } }).catch(err => {console.log(err)})
                }
        },
        created() {
                if( this.$route.query.lat && this.$route.query.lon ) {
                        this.selected.name = this.$route.query.name; this.lat = this.$route.query.lat;
                        this.lon = this.$route.query.lon;
                        this.selectedValue = this.$route.query.name; // value multiselect this.slide =
                        Number(this.$route.query.page); // current slide
                        console.log(this.currentPage) this.axiosWeatherApi();
                        }
                console.log(this.weatherHour, 666);
        }
  }
  </script>
  <!-- Add "scoped" attribute to limit CSS to this component only -->
  <style lang="scss">
  // @import "~vue-multiselect/dist/vue-multiselect.min.css";
  @mixin screen($media) {
     @if $media == 1330 {
        @media (max-width: 1330px) {@content};
     }
  }
  .weather {
        margin-top: 70px;
  }
  h1 {
  }
text-align: center; color: white !important;
font-family: 'Open Sans', sans-serif; font-weight: 400;
  .multiselect { width: 90%; margin: auto;
  }
     .carousel { color: white;
  }
  .carousel-indicators li { outline: none;
  }
  .day {
        color: white;
        font-family: 'Open Sans', sans-serif;
        font-weight: 400;
        font-size: 18px; text-align: center;
        margin-top: 15px !important;
  }
  .VueCarousel-slide { text-align: center;
```

```
}
  .example-slide {
        align-items: center;
        // background-color: #666;
        color: #999; display: flex; font-size: 1.5rem;
        justify-content: center; min-height: 10rem;
        // flex-wrap: wrap;
        @include screen(1330) {
        flex-wrap: wrap;
        }
    }
.example-slide div {
        // width: 100%;
}
.VueCarousel-navigation-prev,
.VueCarousel-navigation-next{
        transition: all 0.2s;
}
.VueCarousel {
        &:hover {
                .VueCarousel-navigation-prev{
                        transform: translateY(-50%) translateX(60%);
                        transition: all 0.2s;
                        &:focus {
                outline: none;
                        }
                }
                .VueCarousel-navigation-next{
                        transform: translateY(-50%) translateX(-75%);
                        transition: all 0.2s;
                        &:focus {
        outline: none;
                        }
                }
        }
}

.VueCarousel-dot {
      &:focus {
                outline: none !important;
        }
}
.VueCarousel-navigation-button{ top: 51.35% !important;
}
.day::first-letter {
        text-transform: uppercase;
}
.nav-carousel-next { width: 20px; height: 20px; border-radius: 20%;
        border-bottom: 5px solid white; border-right: 5px solid white; transform: rotate(-45deg);
}
.nav-carousel-prev { width: 20px; height: 20px; border-radius: 20%;
        border-bottom: 5px solid white;
        border-right: 5px solid white; transform: rotate(135deg);
}
.weather-temp {
        font-family: 'Open Sans', sans-serif ;
}
// ---------- new carousel
.carousel {
```

```
        position: relative;
}
.carousel-inner { position: unset; bottom: 0;
        width: 100%; overflow: unset;
}
.carousel-caption { bottom: 0;
        top: 0;
}
.carousel-control-prev, .carousel-control-next { top: 200px;
}
.carousel-indicators { top: 345px;
        @include screen(1330) { top: 530px;
                margin-bottom: 90px;
        }
}
</style>
          I con.vue
<template>
<divclass="weather-icon-container">
        <div v-if="iconType === 'Atmosphere'" class="icon sun-shower">
                <divclass="cloud"></div>
                <div class="sun">
                        <divclass="rays"></div>
                </div>
                <divclass="rain"></div>
        </div>
        <div v-if="iconType === 'Thunderstorm'" class="icon thunder-storm">
                <divclass="cloud"></div>
                <divclass="lightning">
                        <divclass="bolt"></div>
                        <divclass="bolt"></div>
                </div>
        </div>
        <div v-if="iconType === 'Clouds'" class="icon cloudy">
                <divclass="cloud"></div>
                <divclass="cloud"></div>
        </div>
        <div v-if="iconType === 'Snow'" class="icon flurries">
                <divclass="cloud"></div>
                <div class="snow">
                        <divclass="flake"></div>
                        <divclass="flake"></div>
                </div>
        </div>
        <div v-if="iconType === 'Clear'" class="icon sunny">
                <div class="sun">
                        <divclass="rays"></div>
                </div>
        </div>
        <div
        v-if="iconType === 'Drizzle' || iconType === 'Rain'" class="icon rainy">
                <divclass="cloud"></div>
                <divclass="rain"></div>
        </div>
    </div>
</template>
<script>
export default { components: {}, props: ['iconType']
    ,
```

```
    data () { return {}
    },
    mounted() {
        /* eslint-disable no-console */ console.log(typeof(this.iconType), 111);
    }
}
</script>>
<style scoped lang="scss">
</style>
        ProductA  pp.vue
<template>
    <div class="product-main">
        <div class="nav-bar"></div>
        <div class="product">
            <div class="product-image">
                <img :src="image">
            </div>
            <div class="product-info">
                <h1>{{ title }}</h1>
                <a :href="link" target="_blank">More products like this</a>
                <p v-if="inStock">In stock</p>
                <p v-else :class="{ outOfStock: !inStock }">Out of stock</p>
                <p>Shipping: {{ shipping }}</p>
                <ul>
<li v-for="(detail, idx) in details" v-bind:key="idx">
{{ detail }}
</li>
                </ul>
                <div class="color-sock" v-for="(variant, index) in variants" v-bind:key="variant.variantId"
                :style="{ backgroundColor: variant.variantColor }"
                @mouseover="updateProduct(index)">
                </div>
                <button v-on:click="addToCart"
                :disabled="!inStock"
                :class="{disabledButton: !inStock}">Add to Cart</button>
                <button @click="removeFromCart">Remove from cart</button>
                <ProductTabs :reviews="reviews"/>
            </div>
        </div>
    </div>
</template>
<script>
import ProductTabs from '@/components/Task1/ProductTabs.vue'
import {eventBus} from '../../main.js' export default {
    name: 'ProductApp', components: {
        ProductTabs
    },
    data() { return {
            premium: false, brand: "Vue Mastery", product: 'Socks', selectedVariant: 0,
            link:'https://www.amazon.com/s/ref=nb_sb_noss?url=search-
alias%3Daps&field-keywords=socks',
            details: ['80% cotton', '20% polyester', 'Gender-neutral'],
            variants: [
                {
                    variantId: 2234,
                    variantColor: "green",
                    variantImage:require('../../assets/green.jpg'),
                    variantQuantity: 10
                },
```

```
                {
                    variantId: 2235, variantColor: "blue",
                    variantImage:require('../../assets/blue.jpg'),
                    variantQuantity: 0
                }
            ],
            reviews: []
        }
    },
    methods: {
        addToCart() {
            this.$emit('add-to-cart', this.variants[this.selectedVariant].variantId)
        },
        removeFromCart: function() { this.$emit('remove-from-cart',
this.variants[this.selectedVariant].variantId)
        },
        updateProduct(index) {
            this.selectedVariant = index
        }
    },
    computed: {
        title() {
            return this.brand + ' ' + this.product
        },
        image() {
            returnthis.variants[this.selectedVariant].variantImage
        },
        inStock() {
            returnthis.variants[this.selectedVariant].variantQuantity
        },
        shipping() {
            if (this.premium) {
                return "Free"
            }
            return 2.99
        }
    },
    mounted() {
        eventBus.$on('review-submitted', productReview => {
            this.reviews.push(productReview)
        })
    }
  }
</script>
<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped lang="scss">
    .product { display: flex; flex-flow: wrap; padding: 1rem;
    }
    img {
        border: 1px solid #d8d8d8; width: 70%;
        margin: 15px 40px 40px 40px;
        box-shadow: 0px .5px 1px #d8d8d8;
    }
    .product-image { width: 80%;
    }
    .product-image,
    .product-info {
        margin-top: 10px; width: 45%;
    }
```

```
    .color-box { width: 40px; height: 40px; margin-top: 5px;
    }
    .color-sock { cursor: pointer; width: 40px; height: 40px; margin-top: 5px;
    }
    button {
        margin-top: 30px;
        border: none;
        background-color: #1E95EA;
        color: white;
        height: 40px;
        min-width: 100px;
        margin: 30px 10px 20px 0px; font-size: 14px;
    }
    .disabledButton {
        background-color: #d8d8d8;
    }
    .outOfStock {
        text-decoration: line-through;
    }
</style>
        ProductReview.vue
<template>
    <div>
        <form class="review-form" @submit.prevent="onSubmit">
            <p v-if="errors.length">
                <b>Please correct the following error(s):</b>
                <ul>
}}</li>
        <li v-for="(error, idx) in errors" :key="idx">{{ error
    </ul>
</p>
<p>
 <label for="name">Name:</label>
 <input id="name" v-model="name">
                </p>
<p>
</p>
<p>
<label for="review">Review:</label>
<textarea id="review" v-model="review"></textarea>
<label for="rating">Rating:</label>
<select id="rating" v-model.number="rating">
    <option>5</option>
    <option>4</option>
    <option>3</option>
    <option>2</option>
    <option>1</option>
</select>
                </p>
<p>
 <input type="submit" value="submit">
        </p>
            </form>
            <!-- <input v-model="name"> -->
    </div>
</template>
<script>
import {eventBus} from '../../main.js'
export default {
```

```
    name: 'ProductReview', data () {
        return {
                name: null,
                review: null,
                rating: null, errors: []
        }
    },
    methods: {
            onSubmit() {
                    if(this.name && this.review && this.rating) {
                        let productReview = { name: this.name, review: this.review, rating: this.rating
                        }
                        eventBus.$emit('review-submitted',productReview)
                        this.name = null this.review = null this.rating = null
                    }
                        else {
                                if(!this.name) this.errors.push("Name required.") if(!this.review) this.errors.push("Review
                                required.") if(!this.rating)this.errors.push("Ratingrequired.")
                        }
            }
    }
}
</script>
<style scoped lang="scss">
    .review-form { width: 400px; padding: 20px; margin: 40px;
        border: 1px solid #d8d8d8;
    }
    input {
        width: 100%;
        height: 25px;
        margin-bottom: 20px;
    }

    textarea { width: 100%; height: 60px;
    }

</style>
```