МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ф

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **“ Розробка інформаційної системи аналізу онлайн-платформ для прогнозування метео-даних ”**

Виконав: студент гр. Іт-22сп_____

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

Токар Володимир Васильович_____
(Прізвище та ініціали)

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

Рецензенти: _____
(Прізвище та ініціали)

_____
(Прізвище та ініціали)

ДУБЛЯНИ-2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Освітній ступінь «Бакалавр»
126 – «Інформаційні системи та технології»

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

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Токар Володимир Васильович

1. Тема роботи: «Розробка інформаційної системи аналізу онлайн-платформ для прогнозування метео-даних»

Керівник роботи Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету 30.12.2022 року № 453/к-с.
2. Строк подання студентом роботи 19.06.2023 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, ERD, Static Structure diagram.

4. Зміст розрахунково-пояснювальної записки:

1. *Аналіз передумов застосування іт для управління ризиком у проектах.*
2. *Постановка задачі створення програмного засобу для визначення ризику виробничих умов.*
3. *Проектування інформаційної системи та формування початкових даних.*
4. *Практичне використання розробки.*
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 онлайн метеосервісів; 4 – Аналіз показників для оцінки глобального зондування врожайності культур; 5 – Використання даних супутникового зондування метеоумов для с.г.; 6 – Інформаційна модель метеосервісу; 7 – Діаграми програмної реалізації інформаційної системи метеосервісу; 8 – Фрагменти програмної реалізації ІС метеосервісу; 9 – практичне використання ІС метеосервісу; 10 – 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 управління проектами та безпеки виробництва*		

7. Дата видачі завдання 30.06.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	*30.12.22-01.01.23*	
2.	*Виконання другого розділу та формування головних показників для розрахунків*	*01.01.23-01.02.23*	
3.	*Виконання третього розділу, розрахунків та розробка листів*	*01.02.23-01.03.23*	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	*01.02.23-01.03.23*	
5.	*Вартісне оцінення ефективності пропозицій роботи*	*01.03.23-01.04.23*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	*01.04.23-01.05.23*	
7.	*Завершення роботи в цілому*	*01.05.23-19.06.23*	

Студент _____ Токар В.В.
(підпис)

Керівник роботи _____ Луб П.М.
(підпис)

УДК: 658.51:631.3

Кваліфікаційна робота: 67 с. текст. част., 29 рис., 6 табл., 10 слайдів, 24 джерела.

Розробка інформаційної системи аналізу онлайн-платформ для прогнозування метео-даних. Токар В.В. Кафедра ІТ. – Дубляни, Львівський НУП, 2023.

Виконано аналіз передумов застосування іт для управління ризиком у проектах. Зокрема проаналізовано онлайн метеосервіси, показники оцінки для глобального зондування врожайності культур та наведено недоліки систем метеосервісу, сформульовано задачі кваліфікаційної роботи.

Означено перспективи цифровізації аграрного бізнесу. Описано технології використання даних супутникового зондування метеоумов для сільського господарства.

Описано середовище розробки для програмної реалізації ІС. Виконано проектування ІС та сформовано початкові дані.

Спроектовано інформаційну модель метеосервісу та побудовано діаграми програмної реалізації ІС.

Здіснено програмну реалізацію інформаційної системи.

Наведено приклад практичного використання розробки.

Розроблено заходи із охорони праці та безпеки в надзвичайних ситуаціях.

Ключові слова: інформаційна система, метеосервіс, веб-додаток, Node Package Manager, Vue, Api, прогнозування, дані, сльське 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-спутників спостереження як і будь-які сервіси потребує залучення алгоритмів верифікації та уточнення даних. Саме для таких потреб використовують пости наземного спостереження, тобто – метеостанції.

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 інформаційний сервіс обробки метеоданих для прогнозу погоди і можливості перегляду у погодинному та потижневому форматі. Пошук метеозведення по назві міста, або населеного пункту.

Об’єкт роботи – інформаційні сервіси для відображення метеоданих.

Предмет роботи – показники опрацювання даних метеомоніторингу та їх відображення у програмній реалізації додатку.

Практична цінність – розроблений інформаційний сервіс для перегляду метеоданих, що включає оперативний пошук метеозведеннь за введеним містом/країною, відстежування інформації у погодинному або потижневому форматі та зручний інтерфейс (UserFriendly-Interface). Обране інтегроване середовище розробки (Visual Studio Code). Обраний метод створення додатку (Vue.js Framework + Node Package Manager). Поєднано методи рішення в одну інформаційну систему та розроблене програмне забезпечення для можливості з’ясування погодних умов в конкретному місці. Розроблена інформаційна система протестована.

РОЗДІЛ 1
АНАЛІЗ ПЕРЕДУМОВ ЗАСТОСУВАННЯ ІТ ДЛЯ УПРАВЛІННЯ РИЗИКОМ У ПРОЕКТАХ

1.1. Аналіз онлайн метеосервісів в Україні

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. Такого роду метеосервіс можна зробити в варіанті мобільного додатку [2, 14].

Зазвичай 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.

Метеосервіс (або веб-сайт погоди) – тип веб-сайту, який спеціалізується з метою представлення докладних звіті погоди. 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. При цьому організація діяльності при наявності сьогодні різних форм роботи в дитячому та молодіжному середовищі, таких збір метеоданих, дозволяє збільшити мотивацію участі молодого покоління до прилучення до проектної та дослідницької діяльності.

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) [11].

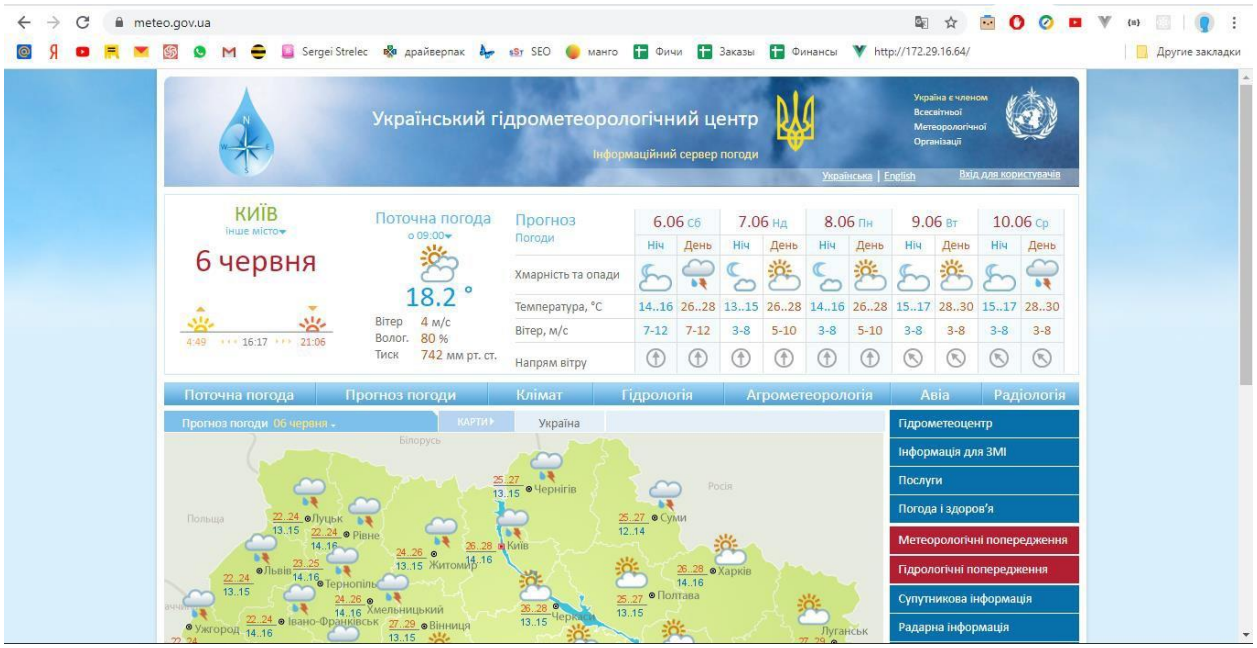

Рисунок 1.1 – Гідрометеорологічний центр

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.) [12].

Рисунок 1.2 – Синоптик із динамічними даинми [12]

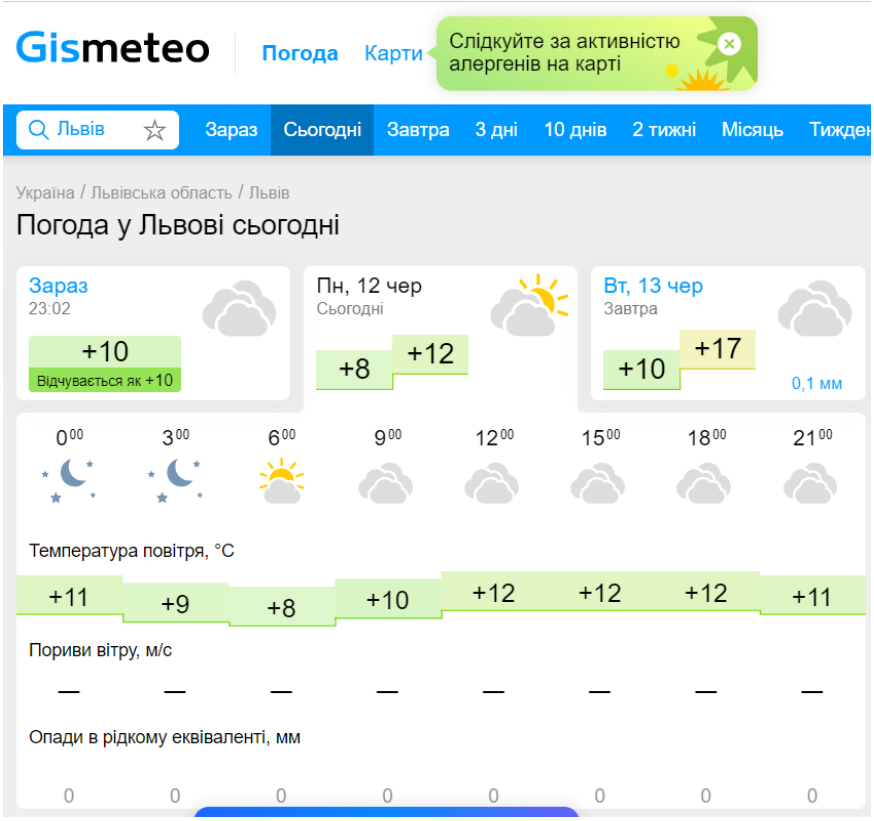

Рисунок 1.3 – Гісметео із динамічними даними [11]

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. Погода Землі функціонує як взаємозалежна система.

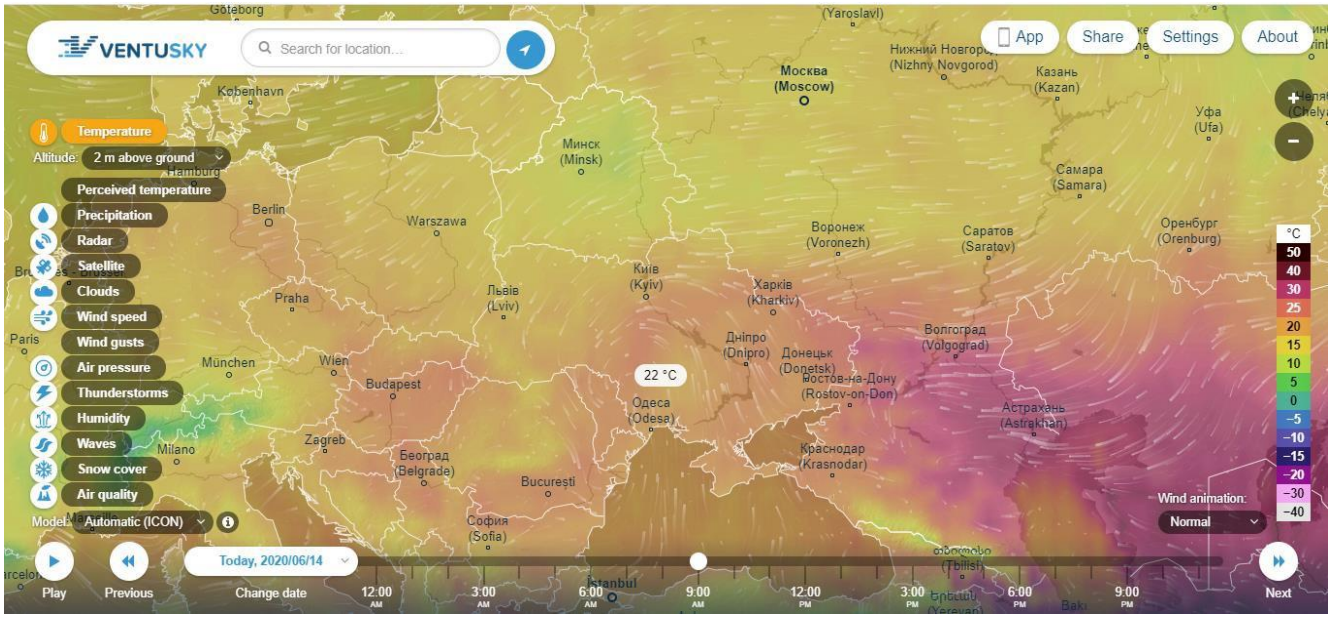

Рисунок 1.4 – Метеосервіс «The Ventusky»

«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 показників оцінки для глобального зондування врожайності культур

Під супутниковим моніторингом в сільському господарстві розуміють діагностику стану посіві на основі зображень високої роздільної здатності, отриманих з використанням методів дистанційного зондування за допомогою штучних супутників Землі. Супутниковий моніторинг в сільському господарстві дозволяє вирішити декілька важливих завдань, які полягають у визначенні стану та однорідності посівів, а також оцінці умов для росту і розвитку рослин, включаючи визначення їх стресового стану. Методи дистанційного зондування за допомогою супутників дають можливість проводити ідентифікацію рослинних утворень, визначати вплив на рослини шкідників, реалізувати інвентаризацію ґрунтів та картографувати динаміку їх змін внаслідок ерозії та вологості, спостерігати за процесами евтрофікації водойм, окреслювати границі рослинних, ґрунтових та водних площ.

Вегетаційний індекс — це показник, що розраховується в результаті операцій із різними спектральними діапазонами даних супутникового дистанційного зондування й має відношення до параметрів рослинності в конкретному пікселі знімка [10]. Ефективність вегетаційних індексів визначається особливостями відбиття та поглинання електромагнітного випромінювання

рослинністю. В даному розділі будуть розглянуті основні вегетаційні індекси, що знайшли широке застосування в області сільського господарства.

NDVI (*англ*. **Normalized Difference Vegetation Index**) — нормалізований диференційний вегетаційний індекс. Найбільш поширений у сільському господарстві, характеризує щільність рослинності й дозволяє аграріям оцінити схожість, ріст, наявність бур'янів або хвороб, а також спрогнозувати продуктивність полів [23].

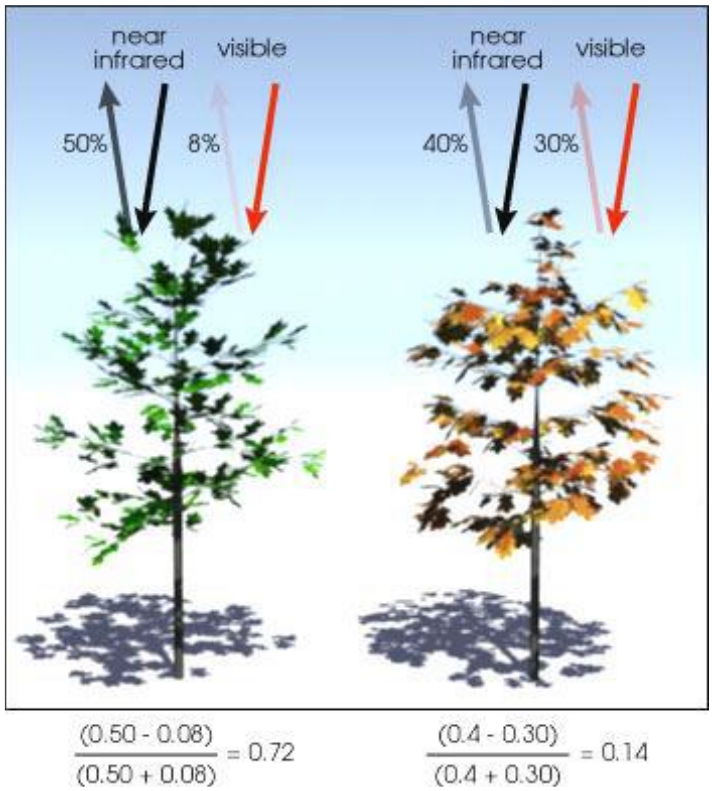

Рисунок 1.5 – Приклад залежності значення індексу NDVI від стану рослинності (Джерело: Robert Simmon, NASA Earth Observatory)

Показники індексу формуються через супутникові знімки зеленої маси, яка поглинає електромагнітне випромінювання у видимому червоному діапазоні та відбиває їх у ближньому інфрачервоному. На червону зону спектра припадає максимум поглинання сонячної радіації хлорофілом, а на ближню інфрачервону зону – максимальне відбиття електромагнітної енергії клітинною структурою листа. Тобто, висока фотосинтетична активність веде до більш низьких значень коефіцієнтів відбиття в червоній зоні спектра і до великих значень у ближній

інфрачервоній зоні. Співвідношення цих показників дозволяє чітко відокремлювати рослинність від інших об’єктів [9, 10].

Окрім безпосереднього використання NDVI в якості показника характеристик рослинності, цей індекс має значну кореляцію з деякими іншими незалежними параметрами: продуктивністю; біомасою; евапотранспірацією; кількістю опадів; вологістю і органічною насиченістю ґрунту; щільністю снігового покриву.

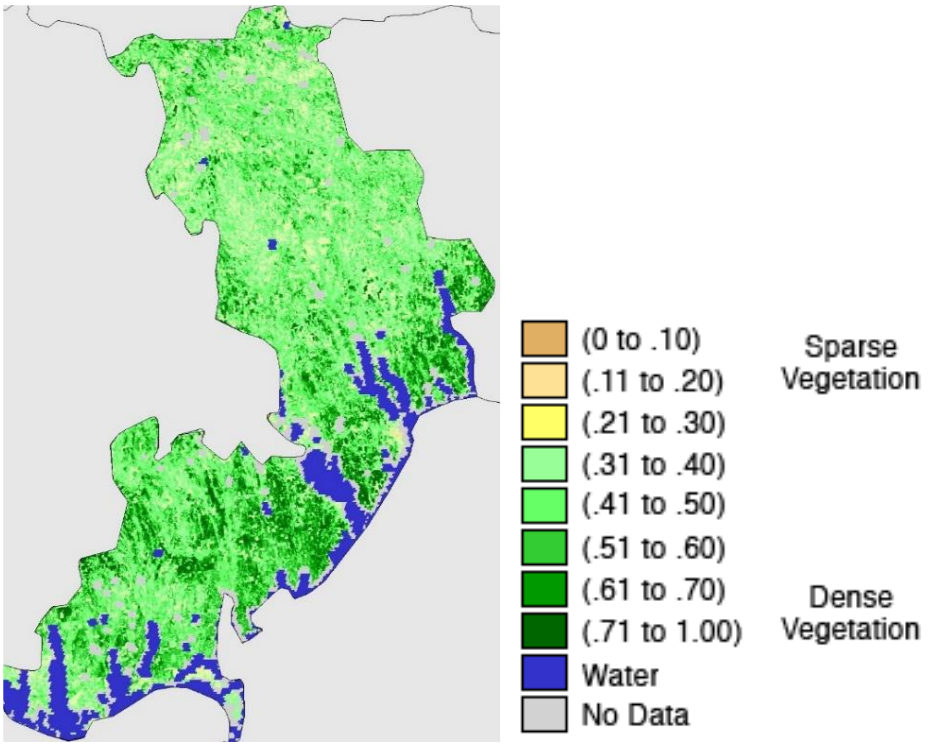

Рисунок 1.6 – Приклад MODIS NDVI (супутник Terra). Одеська область. Період 6-13 квітня 2022 р. (Джерело: GLAM VI Time Series Database) [23]

LAI (*англ*. **Leaf Area Index**) – індекс листової поверхні, [$м^2/м^2$], визначається як загальна площа фотосинтезуючої рослинності на одиницю площі поверхні землі. Він характеризує кількість листового матеріалу в екосистемах і контролює зв’язки між біосферою та атмосферою за допомогою різних процесів, таких як фотосинтез, дихання, транспірація та поглинання дощу (рис. 1.7). Отримані з супутників показники рослинного покрову відповідають загальному зеленому LAI з усіх шарів рослинності, включаючи підстилку, яка може мати дуже важливий внесок, особливо для лісів. Тобто, LAI кількісно характеризує

товщину рослинного покриву.

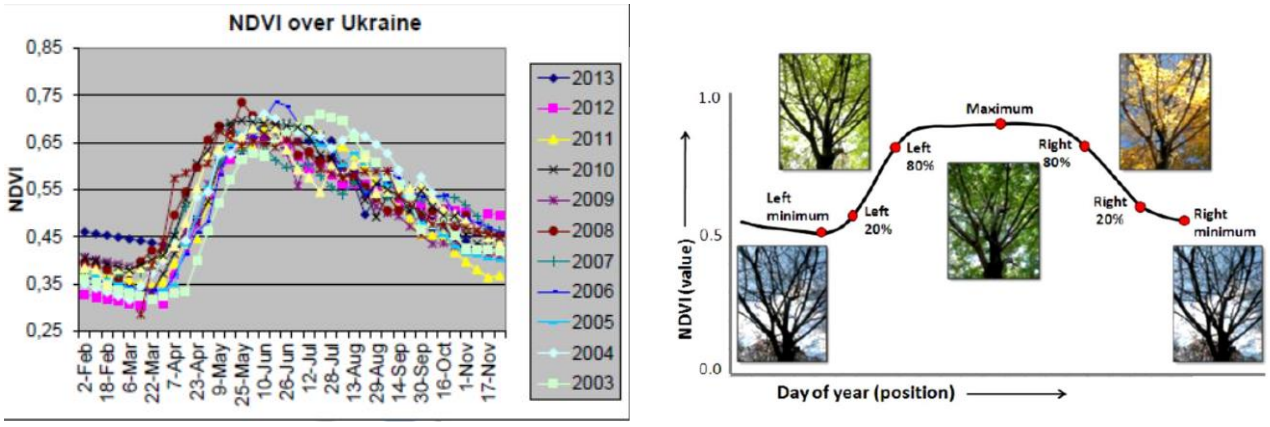

Рисунок 1.7 – Криві часового ходу осереднених за територією України значень NDVI у вегетаційний період з 2021 р. по 2022 р. (а) та схема зміни NDVI по сезонах року (б) [23]

EVI (*англ.* **Enhanced Vegetation Index**) – вдосконалений вегетаційний індекс. Розроблений як поліпшення NDVI шляхом оптимізації сигналу рослинності в областях із високим індексом листової поверхні (LAI). Індекс використовує синю область відображення для корекції фонових сигналів ґрунту і зменшення атмосферних впливів, у тому числі аерозольного розсіювання. Найбільш корисний у регіонах із високим рівнем LAI, тобто у районах з густою рослинністю, де NDVI може перенасичуватися. Значення EVI у пікселах повинні коливатися в діапазоні від 0 до 1. Яскраві об’єкти, такі як хмари й білі будівлі, поряд із темними об’єктами, такими як вода, можуть призвести до аномальних значень пікселів у зображенні EVI. Використовується для оцінки мінливості розвитку культур як в умовах густого рослинного покриву, так і в умовах розрідженої рослинності.

NDWI (*англ.* **Normalized Difference Water Index)** – нормалізований диференційний водний індекс. Відноситься до групи індексів для оцінки вмісту вологи у рослинному покрові, отриманих у вузьких спектральних зонах видимого та ближнього інфрачервоного діапазону. Для розрахунку цього індексу використовують два вузьких спектральних канали, зосереджених на довжинах хвиль близько 0,86 та 1,24 мкм. Абсорбцією рослинною рідиною на каналі 0,86

мкм можна знехтувати, на каналі 1,24 мкм вона є дуже слабкою.

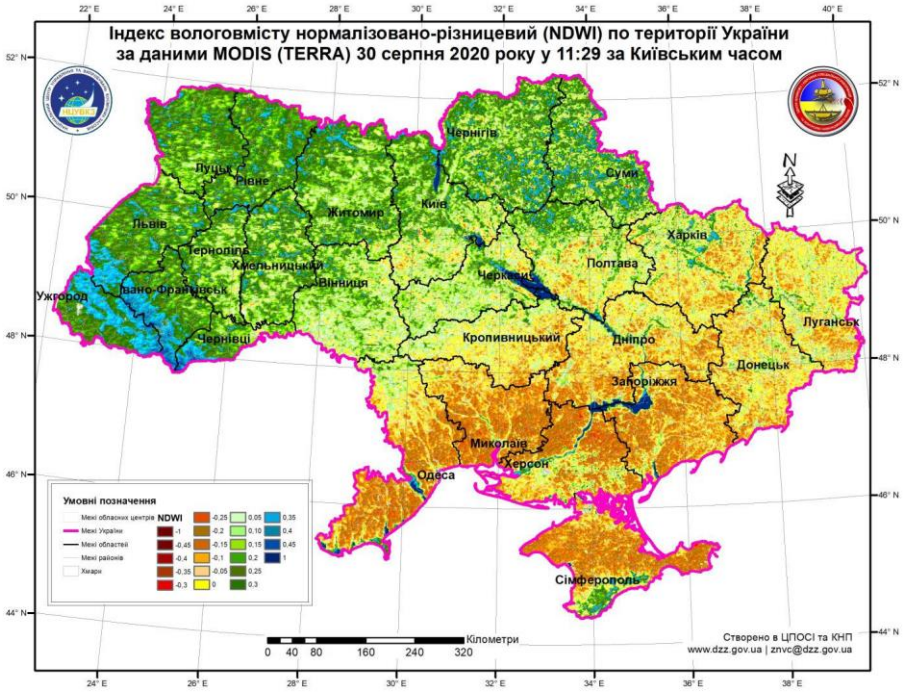

Рисунок 1.8 – Приклад розподілу індексу NDWI по території України (Джерело: Національний центр управління та випробувань космічних засобів) [23]

SIF (*англ.* **Solar Induced Chlorophyll Fluorescence**) – флуоресценція хлорофілу, індукована сонцем. SIF – це електромагнітний сигнал, який випромінює хлорофіл-а асимілюючих рослин. В процесі фотосинтезу частина невикористаної енергії, поглиненої хлорофілом-а, виділяється у вигляді тепла та червоного світіння, тобто SIF (рис. 1.9, а).

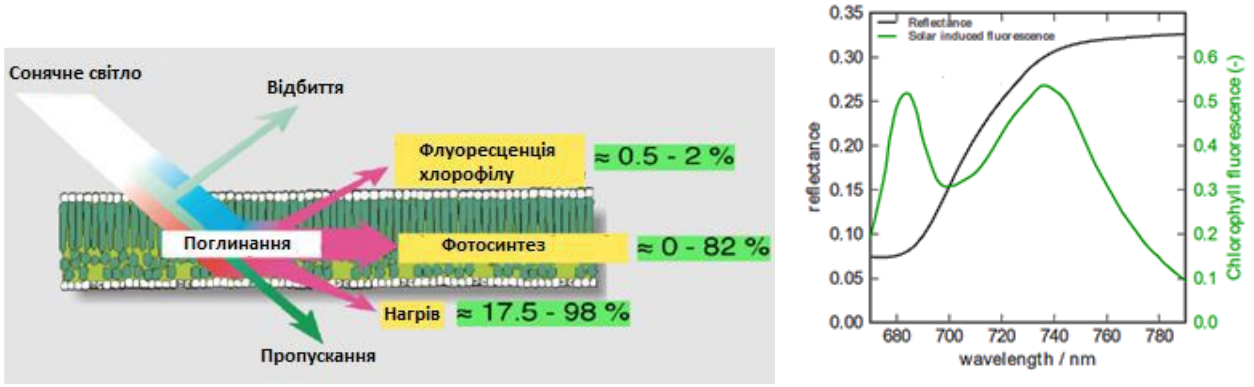

Рисунок 1.9 – Схема перетворення електромагнітного випромінювання сонця на поверхні листа та криві відбиття і флуоресценції хлорофілу в спектральному діапазоні 680-780 нм [23]

Перше зареєстроване спостереження індукованої сонцем флуоресценції було зроблено майже два століття тому Девідом Брюстером, шотландським проповідником, який виявив, що промінь сонячного світла, що потрапляє на зелений спиртовий екстракт лаврового листя, викликає яскраве червоне світіння екстракту. SIF випромінюється у вигляді двопікового спектру, який охоплює діапазон 650–850 нм (рис. 1.9, б).

Випромінювання SIF становить лише невелику частку (зазвичай 0,5-2%) випромінювання у верхній частині рослинного покрову, яке, в основному, складається з відбитого сонячного світла, і його оцінка за допомогою супутникових спектрометрів вимагає як високої спектральної роздільної здатності, так і вдосконаленого схеми пошуку.

1.3. Недоліки систем метеосервісу та постановка задачі

Недоліками сучасних систем реалізації метеосервісів є те, що в основу покладено незручний інтерфейс з інтегрованою рекламою в кожному вільному кутку. Що не дає змоги в повному обсязі передати інформацію про предмет та повністю дослідити його. Також сучасні метеосервіси дають невірні звіти погоди через те, що при розробці були використані старі технології, які не дають змогу інтегрувати сучасне API нових метеоцентрів.

За результатами проведеного аналізу недоліків існуючих віртуальних музеїв поставлені такі задачі:

- вибрати середовище розробки для програмної реалізації інформаційної системи (ІС) у вигляді веб-додатку;
- вибрати сучасні технології розробки додатків;
- вибрати метод створення додатку;
- поєднати вибрані методи рішення в одну інформаційну технологію;
- розробити і протестувати розроблену інформаційну систему.

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ІНФОРМАЦІНОЇ СИСТЕМИ МЕТЕОСЕРВІСУ

2.1. Перспективи цифровізації аграрного бізнесу

Кожна прогресивна країна світу (а яка має статус аграрної – і поготів) запроваджує у сільському господарстві так званий принцип «точного землеробства», тобто, управління кожним квадратним метром землі. Обробіток поля, посів, внесення добрив, боротьба із бур'янами та шкідниками – все це відбувається автоматично, заощаджується кількість посівного матеріалу, добрив, засобів захисту рослин тощо. Передумови комп'ютеризації, яку переживає аграрний сектор, аналогічні іншим ринкам: оптимізація затрат фінансів та часу, підвищення точності розрахунків та планування. Крім того, почали з'являтися програмні комплекси та устаткування для високої швидкості впровадження новацій. Аграрії пізніше за всіх розпочали, але наздоганяють фінансову, промислову та інші галузі економіки.

То ж які ІТ-сервіси сьогодні пропонують в Україні?

AgriChain – це багатомодульна платформа, яка об'єднує рішення для автоматизації бізнес-процесів та ефективного управління всіма напрямками діяльності агропідприємства: управління земельним банком, виробництвом, моніторингом посівів, управління складом, закупівлями і поставками ТМЦ,

Рисунок 2.1. Продукти AgriChain та архітектура ІТ-рішень [15]

контроль роботи техніки і ремонтів, логістики ТМЦ і готової продукції.

Agri Chain вирішує завдання: 1) аудит та управління земельним банком; 2) моніторинг стану посівів; 3) погода, вологість та хімічний аналіз ґрунту; 4) сезонне планування та бюджетування операцій; 5) оперативний облік та планування робіт; 6) ремонти техніки та трансп орту; 7) складські операції; 8) формування первинної документації; 9) відображення операцій в 1С; 10) конструктор бізнес-процесів та звітів.

Компанія **Soft-Farm** створює – комплексне IT-рішення для агровиробників. Об’єднаємо дані з інших систем у єдиний формат та створимо прозору аналітичну систему сільгоспдіяльності для прийняття зважених управлінських рішень.

Рисунок 2.2. Сервіси та агрономічні IT-інструменти Soft-Farm [15]

Сфери діяльності: 1) супутниковий моніторинг; 2) техніка точного висіву та диференційованого внесення; 3) Big Data; 4) автоматизація управлінського обліку; 5) датчики для сільського господарства; 6) агроIT.

Hummingbird Technologies — компанія надає аналітичну інформацію, засновану на даних дистанційного зондування. Обробка даних здійснюється із застосуванням штучного інтелекту і запатентованих алгоритмів обробки зображень. Фірмова експертиза дозволяє нашим клієнтам досягти: підвищення

врожайності, оптимізації внесення добрив і систем захисту рослин, більш успішному веденню сільського господарства і прийняття обміркованих рішень на ранніх етапах [ ].

Рисунок 2.3. Структура дистанційного зондування та IT-послуг Hummingbird Technologies

Сфери діяльності компанії Hummingbird Technologies: 1) БПЛА; 2) супутниковий моніторинг; 3) техніка точного висіву та диференційованого внесення; 4) Big Data. Просування такої діяльності да змогу забезпечувати наступні послуги: 1) оцінення сходів; 2) виявлення бур'янів; 3) планування та корегування норм внесення регуляторів росту рослин, азоту та гербіцидів тощо; 4) картографування ризиків; 5) планування осушувальних робіт.

SAS – сучасні технології для аграрних підприємств, що перетворюють підприємство в сучасний технологічний бізнес []. Зокрема забезпечуються такі моніторингові ресурси: 1) структура посівних площ; 2) моніторинг полів; 3) інформація про поле; 4) журнал агронома; 5) прогноз врожайності; 6) облік земельного банку; 7) пересування техніки; 8) контроль пального; 9) сповіщення про порушення; 10) додаткове обладнання; 11) статус робіт.

Сфери діяльності: 1) БПЛА; 2) супутниковий моніторинг; 3) аналіз ґрунту; 4) датчики для сільського господарства; 5) агро-консалтинг; 6) агроІТ. Послуги

які надає IT-сервіс: 1) автоматизація бізнесу; 2) інтеграція програмних продуктів; 3) послуги агро-консалтингу.

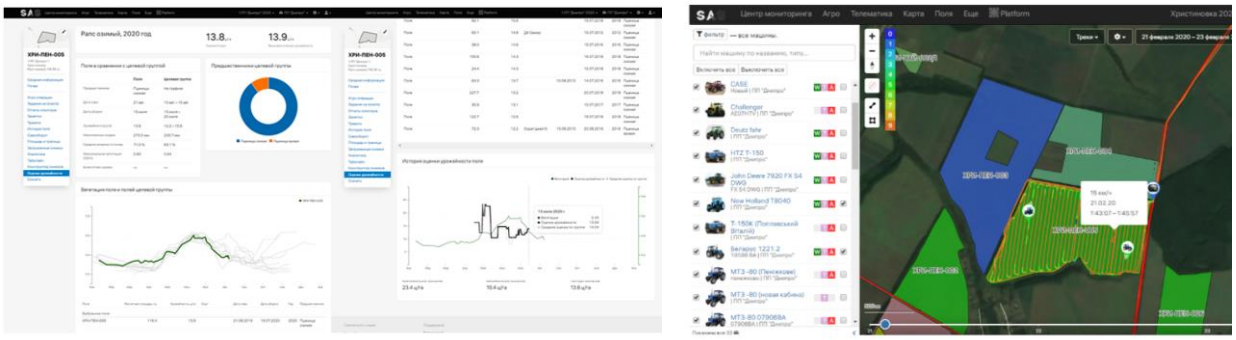

Рисунок 2.4. SAS-технології для аграрних підприємств [15]

DroneUA – це найбільший системний інтегратор безпілотних рішень в Україні. У структурі компанії функціонують власні інженерні та виробничі підрозділи, відкритий центр з обробки даних. DroneUA виступає імпортером і дистриб'ютором найбільш популярних торгових марок в світі дронів.

Рисунок 2.5. Комплекс IT-послуг DroneUA

Сфери діяльності: 1) БПЛА; 2) супутниковий моніторинг; 3) Big Data; 4) АгроІТ. Послуги які надає IT-сервіс – безпілотні технології і комплекси рішень для точного землеробства.

2.2. Використання даних супутникового зондування метеоумов для сільського господарства

Обслуговування сільського господарства має певні особливості, пов'язані з

контролюванням великих територій, які займають сільськогосподарські угіддя. Через нестачу або нерозвиненість мережі наземних пунктів оперативного моніторингу, а також в силу різного роду природних процесів, за яких відбувається постійна зміна меж посівних площ, характеристик ґрунтів і умов вегетації на різних полях, часто не вистачає об'єктивної оперативної інформації, яка необхідна для оцінки поточної ситуації та прогнозування її подальшого розвитку.

Супутникова інформація може допомогти як для вирішення комплексних завдань управління сільськогосподарськими територіями, так й у вузькоспеціалізованих напрямках. За допомогою супутникового моніторингу можна контролювати терміни та якість проведення основних агротехнічних робіт і тим самим оптимізувати управління сільськогосподарським виробництвом (рис.2.6).

З використанням супутникових знімків виконують інвентаризацію і картографування земельних угідь на основі міжнародної класифікації використання земель, а знімки високої роздільної здатності застосовують для створення земельного кадастру.

Систематичні знімки дозволяють проводити спостереження за динамікою розвитку сільськогосподарських культур і прогнозування врожайності. Так, знаючи, як змінюється спектральна яскравість рослинності протягом вегетаційного періоду з урахуванням сільськогосподарського календаря для різних культур, можна по тону зображення полів судити про їх агротехнічний стан і склад культур. В свою чергу, можливість визначення складу культур і площ під ними робить знімки об'єктивним джерелом сільськогосподарської статистики. Крім того, на супутникових знімках відображаються специфічні риси, притаманні тому чи іншому типу ведення сільського господарства, наприклад, зрошення полів.

По знімках високої роздільної здатності виконують оцінку стану посівів: їх однорідності або плямистості, пов'язаної з вимерзанням, вимоканням, вітровим поляганням, впливом шкідників.

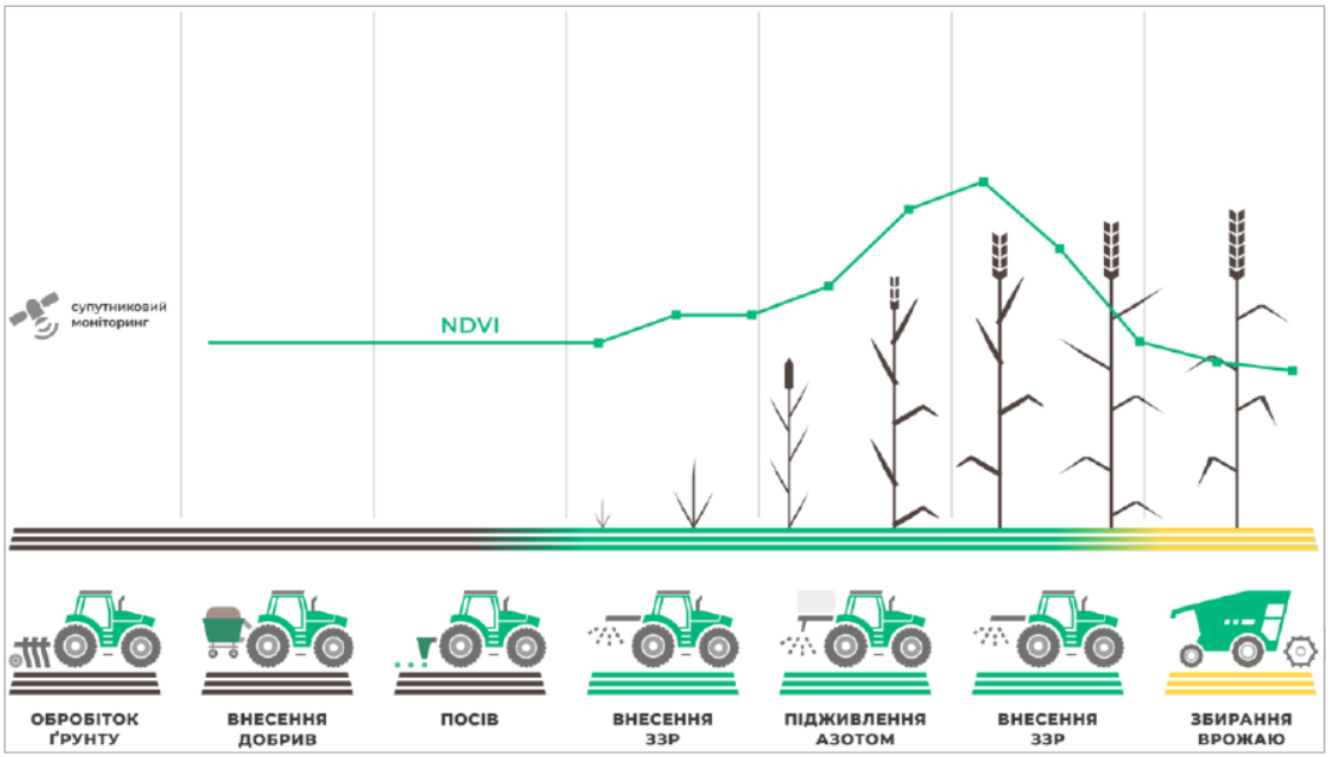

Рисунок 2.6. – Співвідношення часового ходу супутникового вегетаційного індексу NDVI з фазами сільгоспробіт [23]

Виявлення площ, зайнятих основними продовольчими культурами, і оцінка їх розвитку з урахуванням метеорологічних умов надають можливість використання супутникової інформації для прогнозу врожайності.

Зв'язки, встановлені між біомасою рослинності і її спектральною яскравістю, використовуються в складанні карт вегетаційного індексу для оцінки біомаси посівів і пасовищної рослинності (рис. 2.7). Таким чином, застосування методів супутникового дистанційного зондування в сільському господарстві дозволяє вирішувати наступні задачі:

- класифікація типів сільськогосподарських культур;
- оцінка стану посівів (оцінка схожості, зміни фенофаз, розвитку і дозрівання культур);
- визначення областей вимерзання озимих посівів;
- раннє виявлення проявів посухи;
- виділення ділянок ерозії, заболочування, засолення, опустелювання;

 визначення областей загибелі сільськогосподарських культур від хвороб, комах, дефляції, забруднення пестицидами, природних метеорологічних впливів;

 характеристика і стан ґрунтів;

 прогноз врожайності;

 облік та інвентаризація посівних площ;

 моніторинг стану пасовищ, ступеня ураження хворобами і гризунами, зони порушення рослинності в результаті випасу худоби, проективне покриття трав'яною рослинністю;

 стеження за якістю і своєчасністю проведення різних сільськогосподарських заходів;

 загальний моніторинг сільськогосподарської діяльності.

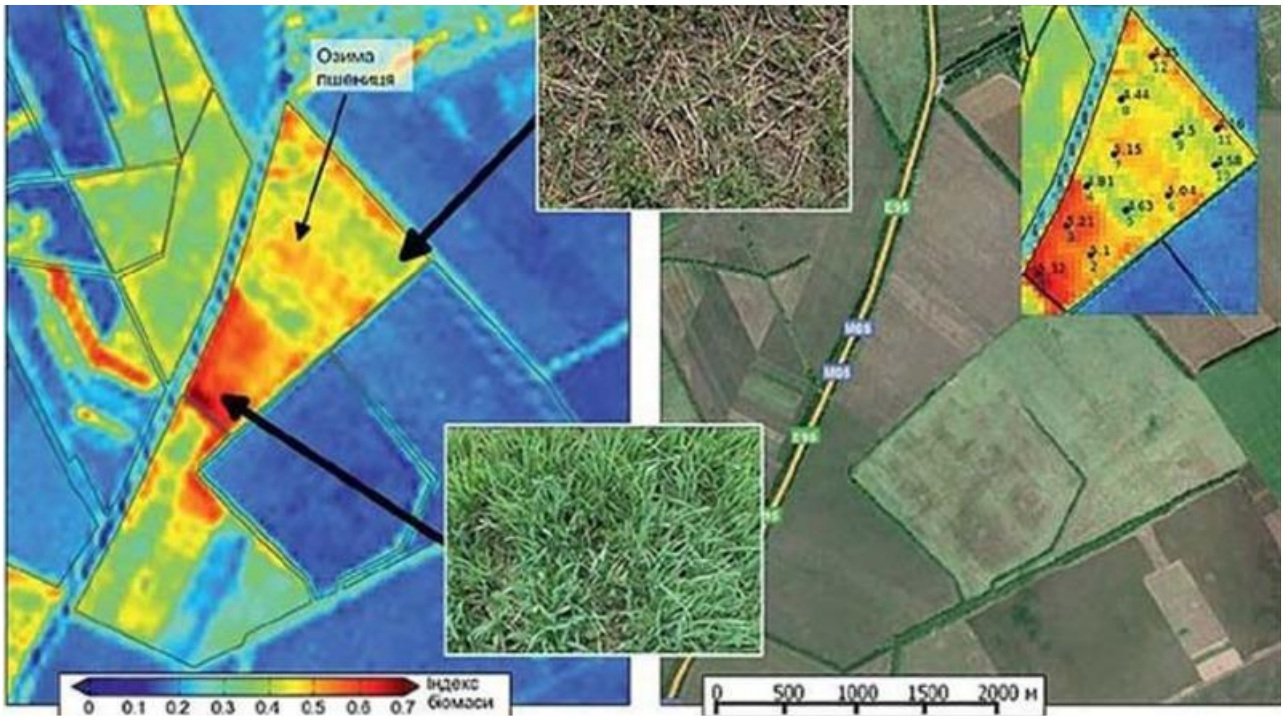

Рисунок 2.7 – Приклад карт оцінки біомаси посівів з платформи EOS Crop *Monitoring* [21]

2.3. Середовище розробки для програмної реалізації ІС

Основним початковим етапом кожної розробки веб-продукту являється вибір середовища розробки.

Для аналізу було підібрані найбільш популярні середовища розробки:

1. VS Code;
2. Net Beans;
3. PyCharm;
4. IntelliJ IDEA;
5. Eclipse;
6. Code: Blocks;
7. Aptana Studio 3;
8. Komodo;
9. RubyMine;
10. X-code.

Вибір кожного з параметрів якості оцінювання базується на стандарті ISO 9126:2001, відповідно кожна з характеристик описується за допомогою кількох вхідних у неї атрибутів.

Для кожного з атрибутів визначається набір метрик, що дозволяють його оцінити. В нашому випадку, це: «Зручність використання», «Продуктивність» або «Часова ефективність», «Функціональність», «інтерфейс програми», «кількість підтримуємих бібліотек», а також параметр – «якість сбірки проекту» (кінцевого зображення після обробки).

Потрібно зазначити, що кожний з наведених критеріїв не є рівнозначним, тому умовно назначаються коефіцієнти важливості кожному з них (табл. 2.1).

Таблиця 2.1 Параметри оцінювання

Параметр	**Індекс**
Якість сбірки проекту	4,0
Кількість підтримуємих бібліотек	2,0
Інтерфейс	1,5
Зручність у використанні	2,0
Часова ефективність	2,5
Функціональність	3,0

Під час рейтингового оцінювання було виставлено оцінки від 1 до 10

(табл.2.2), на основі досвіду роботи з подібними програмами, переглянутих відео, коментарях користувачів даного ПЗ, якості сбірки та інших факторах.

Таблиця 2.2 Таблиця оцінювання середовищ розробки

Назва ПЗ	Якість фінальної обробки	Кількість бібліотек	Зручність у використанні	Часова ефективність	Функціоналність	Інтерфейс
VS Code	9	10	10	8	9	10
Net Beans	8	6	7	8	9	8
PyCharm	4	5	8	7	4	4
Intelli JIDEA	8	9	9	8	8	10
Eclipse	6	7	7	8	5	8
Code:Blocks	4	2	3	6	5	3
Aptana Studio 3	3	2	4	7	4	2
Komodo	7	4	6	5	8	5
Ruby Mine	4	7	4	6	8	5
X-code	6	8	4	7	5	7

Кінцева рейтингова оцінка дорівнює сумі балів за кожний параметр, помножений на відповідний індекс оцінювання. Після обрахунків можна сформувати остаточну рейтингову таблицю (табл. 2.3).

За результатами аналізу було вибрано програмне забезпечення для створення додатку. Обрано VS Code – повнофункціональний професійний застосунок, середовище розробки, система для створення і редагування коду, розроблена компанією Microsoft. Містить найсучасніші засоби для розробників і фахівців в області програмування. Працює в операційних системах Microsoft Windows і Windows NT (як в 32-бітових, так і в 64-бітових).

VS Code використовується для створення веб-додатків, комп'ютерних ігор

тощо.

Таблиця 2.3 Рейтингова таблиця ПЗ

Місце	Назва ПЗ	Оцінка
1	VS Code	137,5
2	IntelliJIDEA	129,5
3	NetBeans	127
4	Eclipse	117,5
5	Xcode	114,5
6	RubyMine	106
7	PyCharm	96
8	Code:Blocks	84
9	Komodo	80
10	Aptana Studio 3	65

Також VS Code має свій власний зневаджувач в реальному часі – це дає змогу розробникам прискорити розробку за рахунок видимості змін при написанні коду. Широкий спектр інтегрованих розширеннь також дозволяє упростити та покращити швидкість розробки.

2.4. Вибір мови програмування для розробки інформаційної системи

Розробка веб-додатків набула великої популярності, а значить зростає набір інструментарію для їх створення. В даному розділі порівняємо (табл. 2.4) та оберемо інструмент для реалізації нашої мети.

Список популярних мов програмування для розробки веб-додатків [2]:

1. PHP;
2. JavaScript;
3. Python;
4. Java;

5. C;
6. C++.

Таблиця 2.4 Таблиця оцінювання мов програмування

Інструмент	**Плюси**	**Мінуси**
PHP	Орієнтація на Web-розробку; Кросплатформеність;	Непослідовний синтаксис; Старі процедурні артефакти
JavaScript	Корисні функціональні налаштування; Постійно підримується та вдосконалюється;	Відсутня типізація даних
Python	Широке застосування; Багато якісних бібліотек; Легкий в налаштуванні	Мала продуктивність та швидкість компіляції; Занадто монолітний;
Java	Розробка через тестування; Гнучкість; Багатопотоковість;	Платне комерційне використання; Низька продуктивність;
C	Продуктивність; Відносна простота мови;	Не використовується в сучасній веб-розробці;
C++	Висока продуктивність; Функціональне програмування;	Низькорівнева мова програмування; Складність знаходження помилок;

Проаналізувавши плюси та мінуси було обрано JavaScript так як ми закриваємо всі наші бажання функціоналом цієї мови.

JavaScript – динамічна скриптова мова програмування що має супутній інструментарій, призначений для розробки додатків у браузерах. Дозволяє користувачам створювати інтерактивний контент за допомогою наявних на ринку інструментів для веб-розробки.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ФОРМУВАННЯ ПОЧАТКОВИХ ДАНИХ

3.1. Вибір *framework* та поєднання рішень в одну інформаційну систему

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. На основі дослідження плюсів та мінусів популярних фреймворків для розробки веб сервісів було обрано Vue фреймворк – програмний каркас з відкритим кодом та контейнери з підтримкою інтеграції сторонніх бібліотек [3, 13, 14, 17].

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 [3, 13, 14, 17]

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) [3].

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.

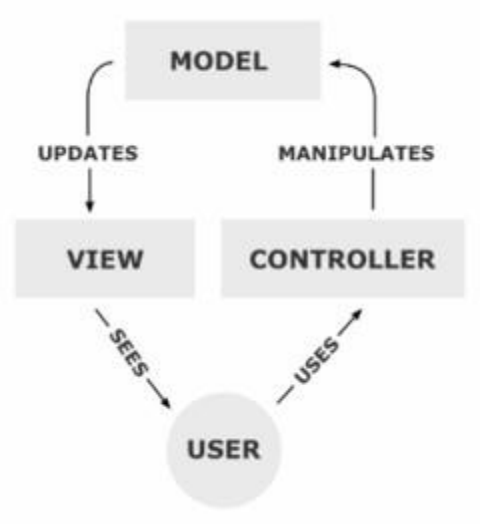

Рисунок 3.1 – Модель MVC

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.

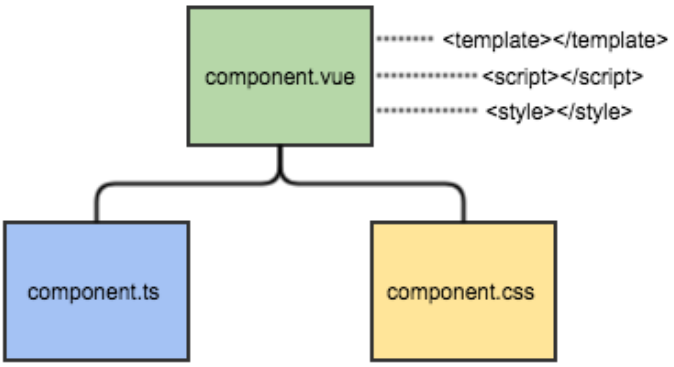

Рисунок 3.2 – Компонентна структура vue.js [17]

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 [3, 17].

Для підключення сторонніх бібліотек будем використовувати пакетний менеджер NPM (Node Package Manager) показаний на рис. 3.3 [3, 17].

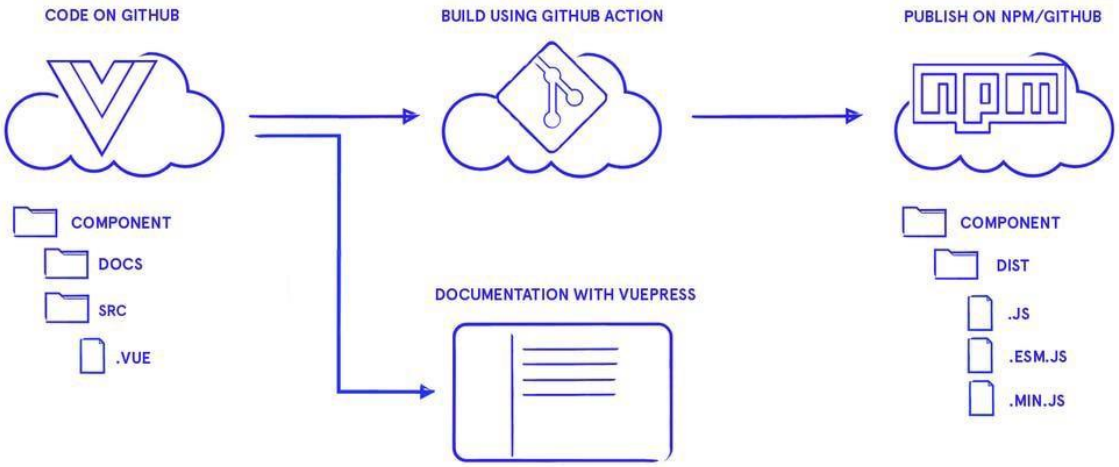

Рисунок 3.3 – NPM (Node Package Manager)

Наприклад, якщо ви хочете, щоб ваш проект використовував Bootstrap, вам необов'язково вручну копіювати всі необхідні файли з сайту Bootstrap і зберігати цю бібліотеку в репозиторії GIT, досить вказати в конфігураційному файлі вашого проекту одним рядком, що вам потрібен Bootstrap і пакетний менеджер npm встановить Bootstrap в копію вашого проекту, при цьому немає потрібності зберігати всі подібні бібліотеки в репозиторії GIT вашого проекту.

Це схоже на Maven для Java або Composer для PHP. Існує два основних інтерфейсу, з якими ви будете взаємодіяти: сайт NPM і набір інструментів командного рядка (CLI).

Зверніть увагу, що NPM поставляється разом з бінарним файлом Node.js, тому вам не потрібно його встановлювати, якщо ви маэте встановлений node.js, проте якщо ви хочете використовувати специфічну версію NPM, ви можете його відновити.

Поєднання вибраних методів рішення в одну інформаційну технологію.

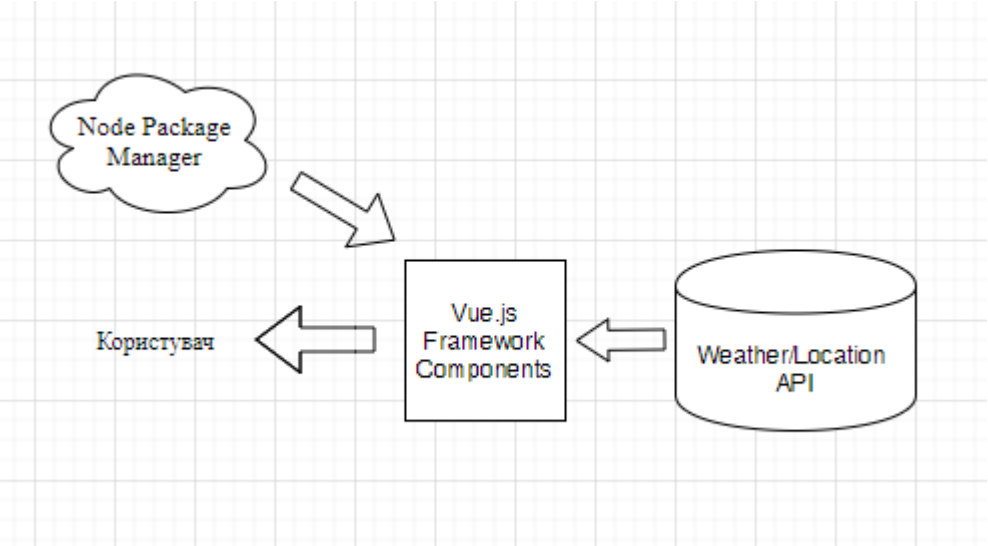

Рисунок 3.4 – Інформаційна технологія для метеосервісу

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

3.2. Інформаційна модель метеосервісу та діаграми програмної реалізації ІС

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 [15, 16, 18].

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.

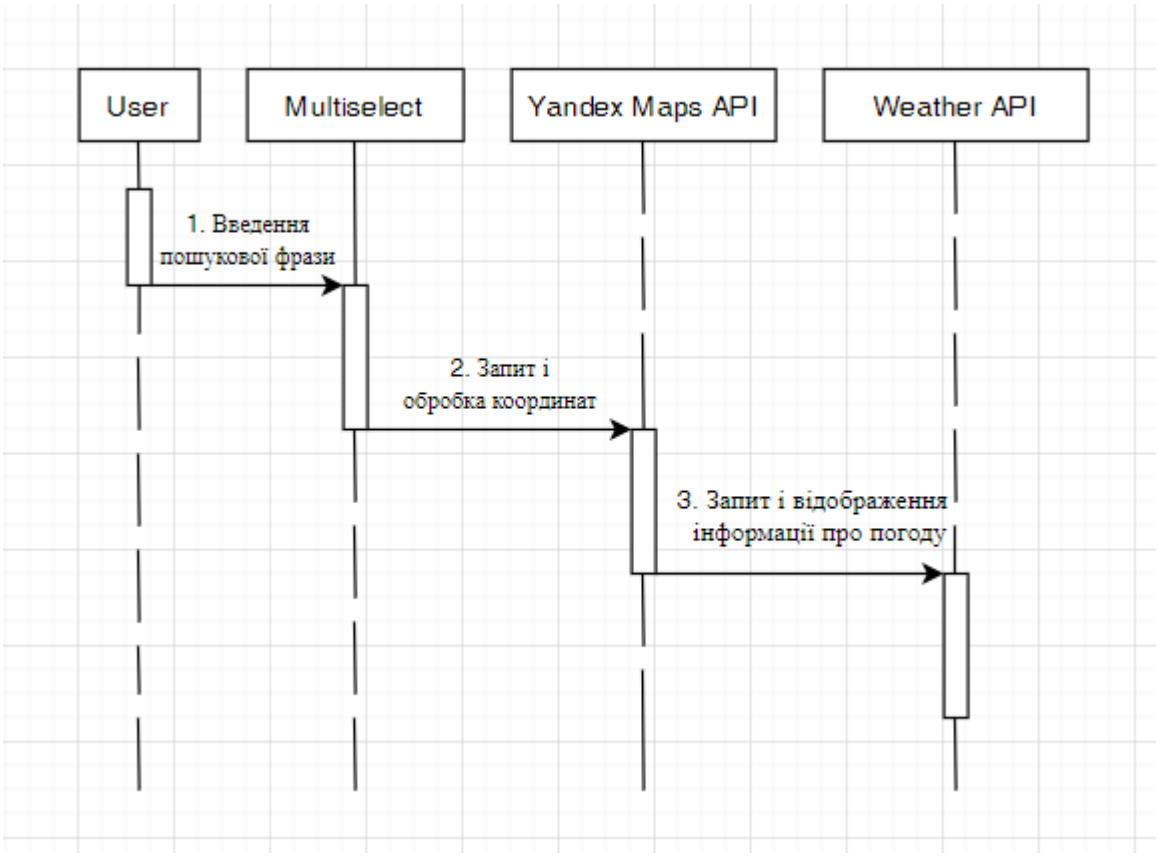

Рисунок 3.5 – Діаграма послідовнестой метеосервісу

Прецедент – можливість модельованої системи (частина її функціональності), завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат [15, 16, 18]. 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.

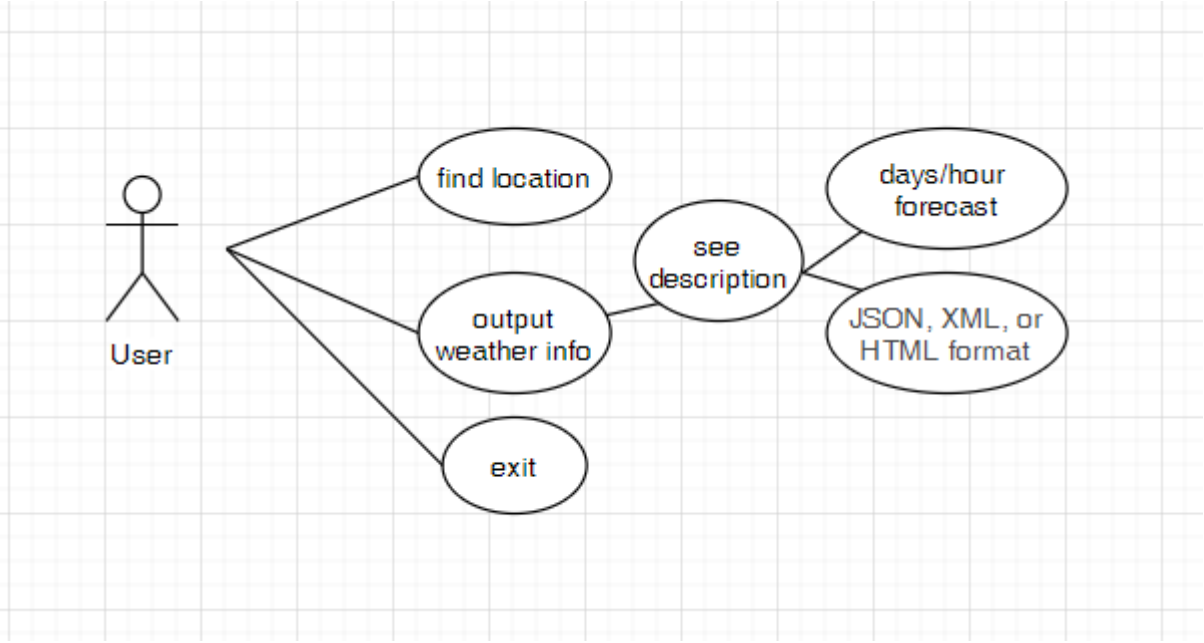

Рисунок 3.6 – Діаграми варіантів використання метеосервісу

ER-модель (від *англ.* Entity-relationship model, модель «сутність - зв'язок») – модель даних, що дозволяє описувати концептуальні схеми предметної області [15, 16, 18]. 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.

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 3.7. 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.

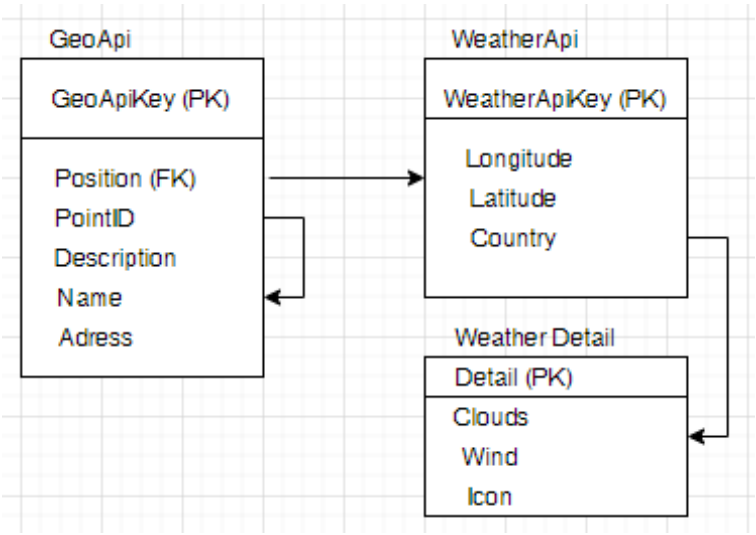

Рисунок 3.7 – ER-діаграма метеосервісу

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 [15, 16, 18]. 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.

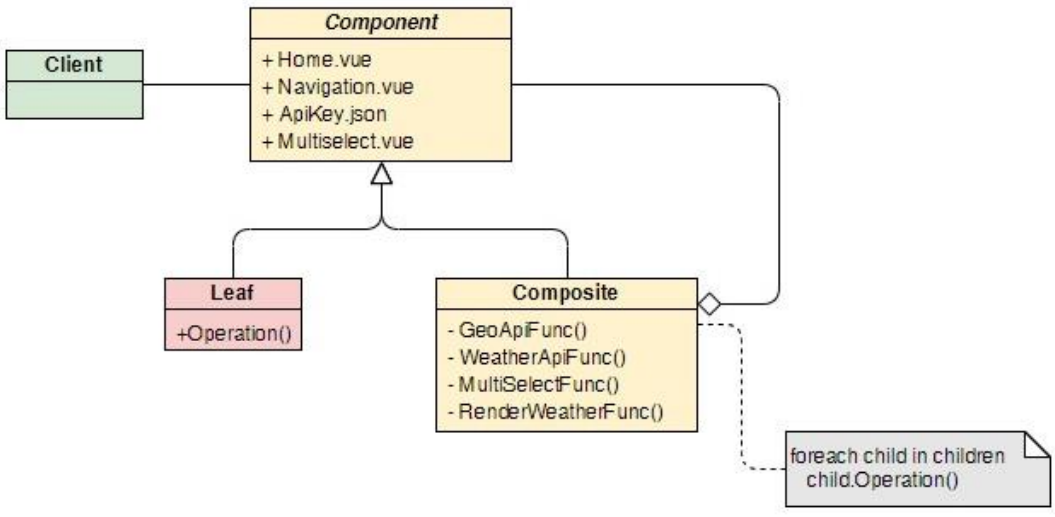

Рисунок 3.8 – Діаграма компонентів

Діаграма компонентів представлена на рис. 3.8. На ній можна виділити:

1. Компоненти, що описують моделі додатку;
2. Сервісний шар додатку;
3. Компоненти контролери;

За представлення відповідають Vue файли.

3.3. Програмна реалізація інформаційної системи

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. 3.9.

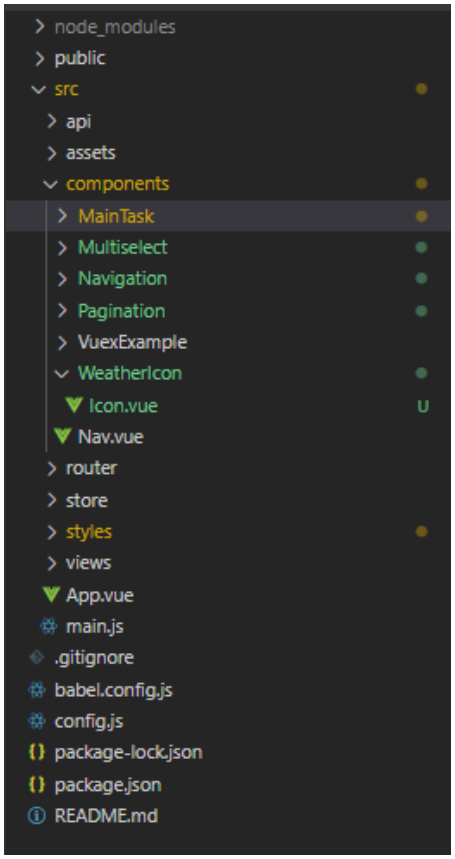

Рисунок 3.9 – Структура проекту ІС метеосервісу

Проект метеосервісу складається з 5-ти основних папок:

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
```

Після ініціалізації контейнеру станів Vuex потрібно загрузити контекст. Потрібно розбити контейнер на модулі для кожного логічного стану компонентів та написати для кожного модуля слухачі Getters, Actions та Mutations, як показано нижче:

```
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
}
```

В теці styles містяться конфігураційні файли для налаштування стилів нашого сервісу (рис. 3.10).

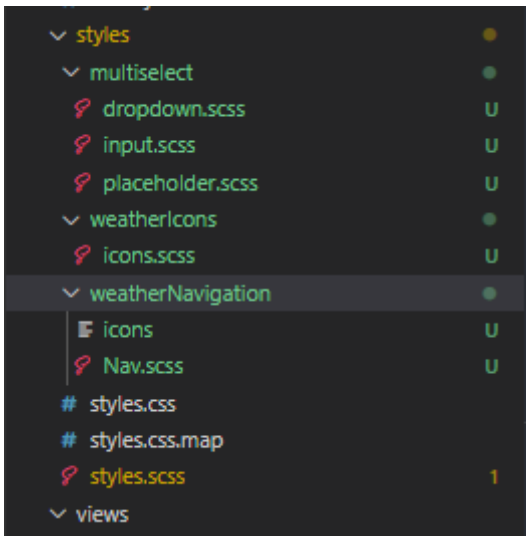

Рисунок 3.10 – Структура стилів у програмній реалізації ІС

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.

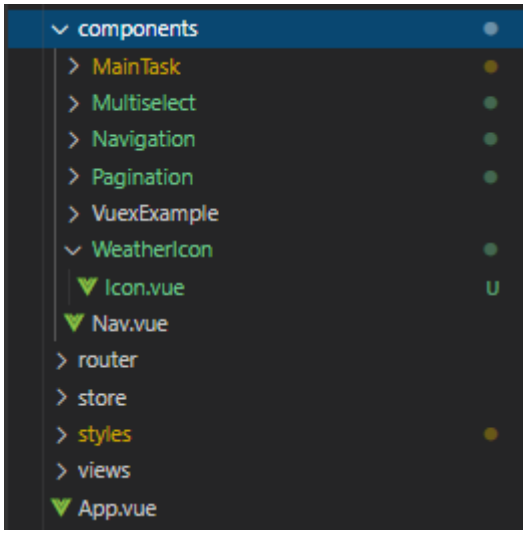

Рисунок 3.11 – Структура компонентів у програмній реалізації ІС

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

Приклад двух головних методів, які звертаючись до гео-API, обмінюються параметрами та зберігають інформацію звіту погоди, яку користувач хоче побачити:

```
getGeoApi(currentValue) {
   this.temp = [];
   this.time = [];
if (currentValue.length <= 2)
{this.options = []; this.temp = [];
}
else
{
this.$axios.get(`${config.geoApi}?apikey=${config.apiKeyGeo}&format=
json&geo code=${currentValue}`)
   .then( response => {
   console.log(response)
   this.geo =
   response.data.response.GeoObjectCollection.featureMember;
   this.geo.forEach(obj => {
   this.options.push(obj.GeoObject); // витягнути всі обєкти з
   масива api
this.coord.push(obj.GeoObject.Point.pos); // витягнути координати
   });
   console.log(this.options)
  })
```

```
 }
},
axiosWeatherApi() {
  var coord = currentValue.Point.pos.split(" ");
  this.lon = coord[0];
  this.lat = coord[1]; this.axiosWeatherApi();
  this.$axios.get(`${config.weatherApi}lat=${this.lat}&lon=
  ${this.lon}&units=metric&appid=${config.apiKeyWeather}`)
    .then( response => {
       console.log(response)
       this.weather = response.data.list;
        this.weather.forEach(obj => { this.temp.push(obj.main.temp);
           this.time.push(obj.dt_txt);
        });
     this.displayWeather(this.weather);
   })
}
```

За допомогою методу displayWeather() відбуває рендер інтерфейсу перегляду 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)})
}
```

РОЗДІЛ 4
ПРАКТИЧНЕ ВИКОРИСТАННЯ РОЗРОБКИ

4.1. Результати та головні кроки із практичного використання метеосервісу

Робота з додатком в головному починається з пошуку місця в котрому користувач хоче знати погодні умови. Пошук міста чи країни (рис. 4.1) відбувається в мультиселект.

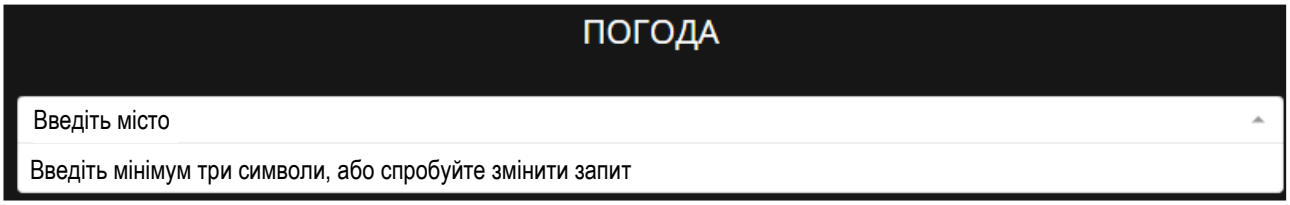

Рисунок 4.1 – Пошук місця, у якому користувач бажає дізнатись погоду

Після того, як користувач почав вводити данні у поле запиту – метеосервіс робить запит до гео-API, та показує весь можливий список регіонів, які задовольняють умови запиту (рис. 4.2).

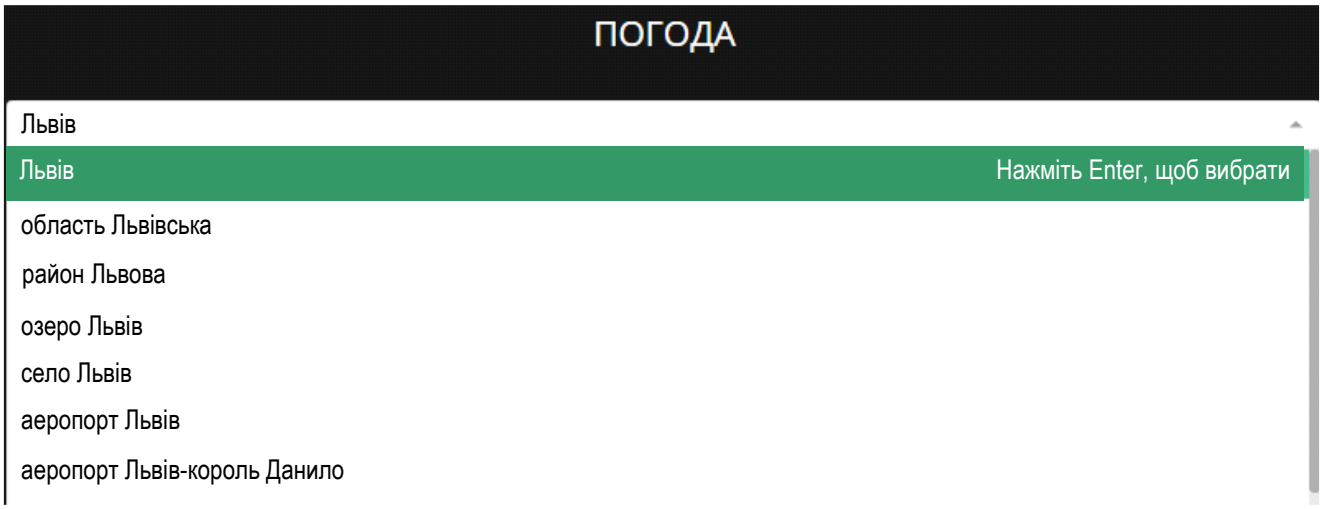

Рисунок 4.2 – Введення запиту в поле

Наступним кроком є виведення інформації по погоді в інтерфейс. У користувача є можливість перегляду даних звіту погоди у погодинному форматі, кожні 3 години в 5-ти наступних дні. Знизу знаходиться панель пагінації –

переходу до наступного дня. Також при успішному запиті в url браузера зберігаються параметри Name – назва міста, Lat/Lon – координати довготи та широти, Page – відкрита сторінка. Відображення головної інформації на рис. 4.3 відбувається на самописному слайдері.

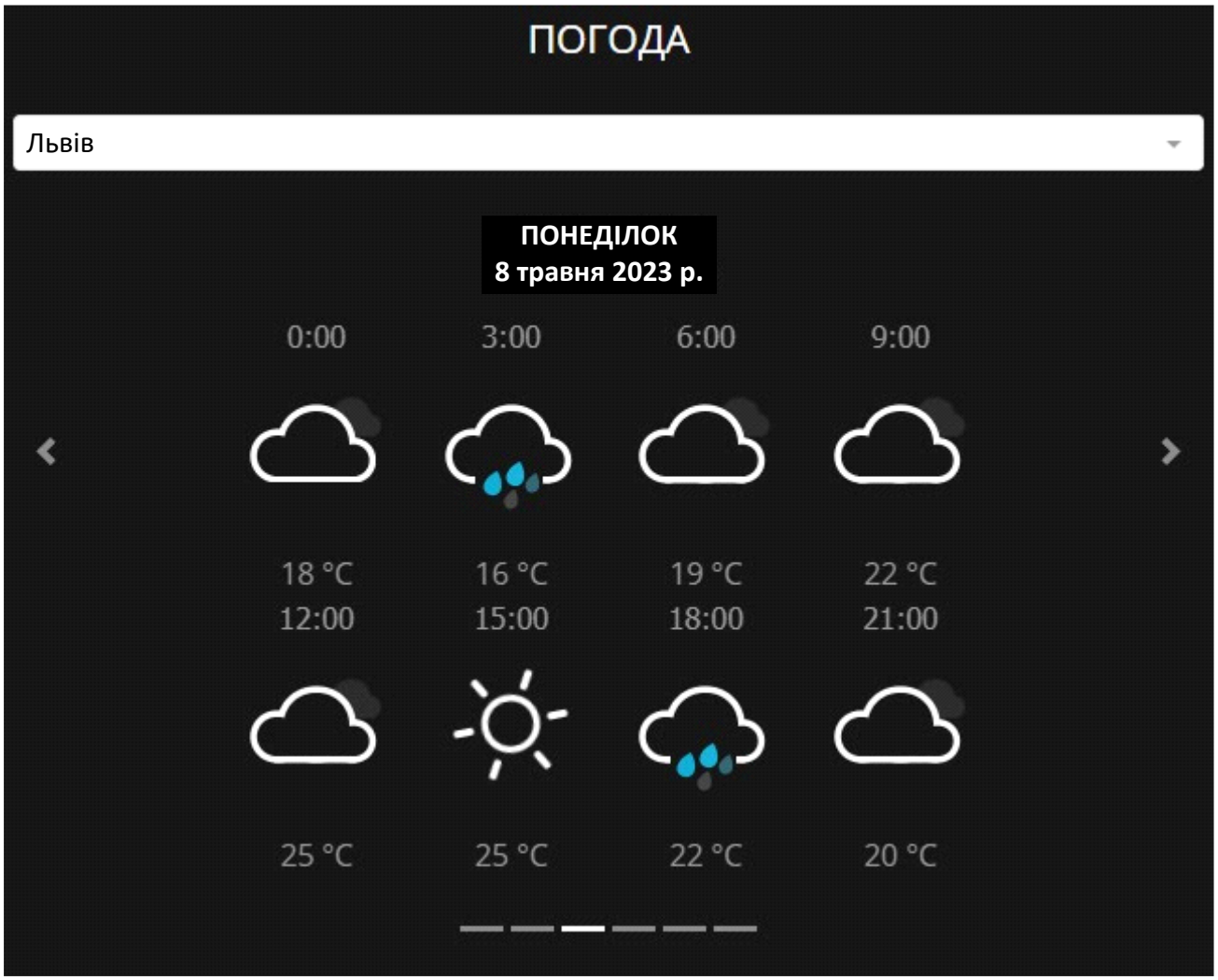

Рисунок 4.3 – Відображення звіту погоди

Перевіривши функціонування основних елементів веб додатку можна зробити висновок, що система працює нормально. Відображення інформації про погоду та самі запити до API метеостанцій здійсюються корректно. Система валідації та захисту працює в визначених межах.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Структурно-функціональний аналіз технологічного процесу

Розробка та вживання ефективних заходів запобігання аварійним і травмонебезпечних ситуаціям можливі лише при завчасному виявлення тих небезпек, з яких починаються процеси їх формування. Оскільки небезпечні умови не завжди завчасно можна виявити, а для вивчення небезпечних дій іноді потрібно багато часу, щоб зібрати статичний матеріал, то і методи виявлення цих небезпек повинні бути відповідно диференційовані (табл. 5.1) [9].

Таблиця 5.1. Моделі формування та виникнення травмонебезпечних і аварійних ситуацій

Вид робіт, виробничий підрозділ, робоче місце, виробниче обладнання, склад агрегату	Виробнича небезпека			Можливі наслідки	Заходи запобігання небезпечним ситуаціям
	Небезпечна умова (НУ)	Небезпечна дія (НД)	Небезпечна ситуація (НС)		
Виконання робіт із електрообладнанням	Не вимкнено живлення. Відсутність заземлення.	Нехтування правилами ТБ	Ураження струмом	Травма (Т)	Проведення повторного інструктажу з ТБ. Розробка нових способів захисту. Встановлення заземлення.
НД НУ → НС → Т					

Відповідно до аналізу небезпечних умов, які існують у виробничому процесі виокремлено такі наступні за характером дії на працівника їх групи [9]:

– характеризують стан або рівень небезпеки обладнання, які використовуються.

– сприяють виникненню технологічних помилок обслуговуючого персоналу впродовж виробничого процесу;

– створювати умови та можливість проникнення працівника в небезпечну зону;

– приводять до виникнення небезпечних дій (внаслідок низького рівня професійної підготовки працівників та організації навчання з охорони праці).

Моделі формування та виникнення травмонебезпечних і аварійних ситуацій в комп'ютерному кабінеті представлено у вигляді моделі формування та виникнення травмонебезпечних і аварійних ситуацій – табл. 5.1.

5.2. Розрахунок освітлення приміщення комп'ютерного кабінету

Освітленість виробничих приміщень може бути штучною і природною. Природне освітлення при правильному обладнанні найбільш сприятливе для людини. Основні вимоги для освітлення наступні:

– освітлення повинне бути достатнім для швидкого і легкого розпізнання об'єктів роботи;

– освітлення повинно бути рівномірне без різких тіней;

– джерело світла не повинно осліплювати працівника;

– рівень освітленості не повинен обмежуватись часом.

Природне освітлення забезпечується обладнанням вікон (бокове освітлення) фонарів і світильних покрить приміщень (верхнє освітлення). Природне освітлення нормується коефіцієнтом природної освітленості. Коефіцієнт природної освітленості – це процентне відношення фактичної освітленості $F_в$ в будь-якій точці приміщення до освітленості $F_н$ розсіяної світлом небозводу точки, яка лежить на відкритій місцевості. Розрахунок природного освітлення через бокові вікна по нормам освітленості ведеться для самої дальньої від вікон точки, тобто знаходять мінімальне значення стик коефіцієнта природної освітленості [9]:

$$e_{\min} = \frac{Fb}{Fн} \cdot 100. \qquad (5.1)$$

Значення коефіцієнта природної освітленості визначається не менше чим в п'яти точках. Значення коефіцієнта природної освітленості для сільськогосподарських виробничих приміщень в даному випадку ремонтній майстерні, беремо $e_{min} = 5\%$.

Розрахунок природного освітлення зводиться до визначення площі світлових променів.

Сумарну площу світлових променів $\sum F_o\,(м^2)$ по коефіцієнту природної освітленості для бокових променів визначаємо по формулі:

$$\sum F_o = \frac{F_н \cdot e_{\min} \cdot r_o \cdot K}{100 \cdot \tau \cdot Г_1}, \qquad (5.2)$$

де $F_н$ – площа підлоги, м^2; e_{min} – величина мінімального коефіцієнта природного освітленості; τ – загальний коефіцієнт світловикористання віконного отвору із врахуванням його забруднення, τ = 0,25; r_o – світлова характеристика вікна, r_o – 9,5; $Г_1$ – коефіцієнт, який враховує підвищення освітленості за рахунок світла, яке відбивається від стін і стелі, $Г_1$ = 1,2; K – коефіцієнт, який враховує затінення вікон сусідніми приміщеннями і загорожею, K = 1.

$$\sum F_o = \frac{36 \cdot 0,5 \cdot 9,5 \cdot 1}{100 \cdot 0,25 \cdot 1,2} = 5,7 м^2.$$

Кількість світлових променів визначимо:

$$N = \frac{\sum F_o}{F_o}, \qquad (5.3)$$

де F_o – площа вікна згідно стандарту, м^2.

$$N = \frac{5,7}{6} = 0,95.$$

Приймаємо кількість вікон – одне вікно.

При розрахунку природного освітлення найбільш поширеним і простим є метод світлового потоку. При цьому методі розраховуємо світловий потік $F_л$ (Лк), який повинна випромінювати кожна лампа (при заданій кількості ламп).

$$F_л = \frac{k \cdot S_п \cdot E}{n_л \cdot \eta \cdot r^2}, \qquad (5.4)$$

де k – коефіцієнт запасу, $k = 1{,}3$; $S_п$ – площа підлоги, м^2; $S_п = 36$м^2. E – нормативна освітленість, Е = 300 Лк; $n_л$ – кількість встановлених ламп, $n_л$=6 од; η – коефіцієнт використання світлового потоку, $\eta = 0{,}25$; r – коефіцієнт нерівномірності освітленості, $r = 0{,}545$.

Коефіцієнт запасу (K) враховує можливість забруднення світильників пилом, що залежить від характеру виробництва.

Розрахунок штучного освітлення починаємо із визначення висоти розташування світильника і їх кількості. Висоту $h_н$ (м) розташування світильників над робочим місцем знаходимо за формулою:

$$h_н = H - (h_1 + h_2), \qquad (5.5)$$

де H – висота приміщення, м; h_1 – віддаль від підлоги до освітлювальної поверхні, м; h_2 – віддаль від стелі до світильника, м.

$$h_н = 4{,}5 - (2{,}2 + 1{,}5) = 0{,}8\text{м}.$$

При симетричному розміщенні світильників по вершинах квадратів їх кількість визначається за формулою:

$$n_c = \frac{S_п}{l^2}, \qquad (5.6)$$

де l – віддаль між світильниками, м.

Підставивши значення отримаємо:

$$n_c = \frac{36}{9} = 4 \text{ од.}$$

Тоді світловий потік буде становити

$$F_л = \frac{1{,}3 \cdot 36 \cdot 300}{4 \cdot 0{,}25 \cdot 0{,}545} = 2576{,}2 \text{ Лк}.$$

При світловому потоці 2576,2 Лк для заданої лампи вибираємо тип і потужність.

Вибираємо тип лампи – люмінесцентну, потужністю 40Вт.

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

2. Сучасні технології дистанційного зондування Землі та апаратура, встановлена на супутниках, дозволяють отримати не тільки зображення, але й широкий набір різноманітних цифрових даних, які можуть бути використані для оцінки потенційної врожайності сільськогосподарських культур та розробки заходів щодо запобігання негативних тенденцій у їх розвитку. Для цього використовують вегетаційний індекс що являє собою показник, який розраховується в результаті операцій із різними спектральними діапазонами даних супутникового дистанційного зондування й має відношення до параметрів рослинності в конкретному пікселі знімка

3. Проаналізовані діючі інформаційні системи метеосервісу дали змогу виявити їх позитивні сторонни та охарактеризувати інормаційне наповення сайтів. На цій основі окреслено завдання щодо розробки веб-орієнтованого додатку для опрацювання метеоданих.

4. Обране середовище розробки для створення метеосервісу. Після аналізу недоліків та переваг сучасних програм для реалізації метеосервісу вибір впав на програму VS Code, що дозволяє створювати кросплатформенні та хмарні застосунки.

5. Обрана мова програмування JavaScript що поєднується із фреймворком Vue.js. Застосовуючи сучасні рішення від Vue забезпечено досягнення належних показників продуктивності та швидкості роботи додатку.

6. 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.

7. 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).

8. Реалізоване програмне забезпечення для функціонування метеосервісу показало свою валідність, це також підтверджено результататми тестування розробленої інформаційної системи. Поєднавши разом інструменти створення веб-додатків отримано інформаційну технологію – метеосервіс, що дозволяє якісно представити звітну інформацію стосовно погодих умов, що вибрано користувачем.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. 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.

2. 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.

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

4. 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

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

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

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

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

9. 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.

10. Лихочвор В.В. Рослинництво. Технології вирощування сільськогосподарських культур. Львів: НВФ "Українські технології", 2002. 800 с.

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

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

13. Резіг Д. Секрети JavaScript / Резіг Д., Марас І., Бібо Б. К.: Вільямс, 2017. 544 с.

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

15. 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.

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

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

18. 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.

19. Шилдт Г. C# 4.0: полное руководство / Г. Шилдт. – М.: ООО “И.Д. Вильямс”, 2011. 1056 с.

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

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

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

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

24. Vue: як використати компоненти. URL: https://medium.com/@modex13/vue-js-2-8f029ba5a60c

ДОДАТКИ

Додаток А.

Фрагмент коду програмної реалізації 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>
```

Icon.vue

```
<template>
<div class="weather-icon-container">
    <div v-if="iconType === 'Atmosphere'" class="icon sun-shower">
        <div class="cloud"></div>
        <div class="sun">
            <div class="rays"></div>
        </div>
        <div class="rain"></div>
    </div>
    <div v-if="iconType === 'Thunderstorm'" class="icon thunder-storm">
        <div class="cloud"></div>
        <div class="lightning">
            <div class="bolt"></div>
            <div class="bolt"></div>
        </div>
    </div>
    <div v-if="iconType === 'Clouds'" class="icon cloudy">
        <div class="cloud"></div>
        <div class="cloud"></div>
    </div>
    <div v-if="iconType === 'Snow'" class="icon flurries">
        <div class="cloud"></div>
        <div class="snow">
            <div class="flake"></div>
            <div class="flake"></div>
        </div>
    </div>
    <div v-if="iconType === 'Clear'" class="icon sunny">
        <div class="sun">
            <div class="rays"></div>
        </div>
    </div>
    <div
    v-if="iconType === 'Drizzle' || iconType === 'Rain'" class="icon rainy">
        <div class="cloud"></div>
        <div class="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>
        <formclass="review-form" @submit.prevent="onSubmit">
            <pv-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>
<labelfor="name">Name:</label>
<input id="name" v-model="name">
                </p>
<p>
</p>
<p>
<labelfor="review">Review:</label>
<textarea id="review" v-model="review"></textarea>
<labelfor="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>
```

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.js

```
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.js
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
}
```