

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти (освітній ступінь))

на тему СИСТЕМА ПЛАНУВАННЯ І АНАЛІЗУ ДЛЯ ОТРИМАННЯ
МАКСИМАЛЬНОГО ВРОЖАЮ З ПЕВНОЇ ТЕРИТОРІЇ

Виконав: студент 2 курсу, групи КНТ-612м
Спеціальності

123 «Комп'ютерна інженерія»

(код і назва спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

ЧЕРНОВ Д. К.

(прізвище та ініціали)

Керівник ІЛ'ЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент КАПЛІЄНКО Т.І.

(прізвище та ініціали)

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

Факультет комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) магістерський
 Спеціальність 123 Комп'ютерна інженерія
 (код і назва)

Освітня програма (спеціалізація) Спеціалізовані комп'ютерні системи
 (назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«_____» _____ 20__ року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЧЕРНОВ Дмитро Костянтинович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Система планування і аналізу для отримання максимального врожаю з певної території

керівник проекту (роботи) ІЛЬЯШЕНКО Матвій Борисович, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проекту (роботи) 11 грудня 2023 року

3. Вихідні дані до проекту (роботи) База даних, нейромережа, карти ґрунту у відкритому доступі

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

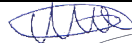
1) Аналіз способів визначення придатності земельної ділянки

2) Вибір технологій для розробки системи;

3) Проектування системи планування і аналізу для отримання максимального врожаю

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б., к. т. н., доцент		
Нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 10.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	10.10.2023 р.	
2	Аналіз способів визначення якості ґрунту і дослідження наявних рішень	16.10.2023 р.	
3	Вибір технологій для розробки системи	20.10.2023 р.	
4	Проектування бази даних	24.10.2023 р.	
5	Побудова логічної моделі бази даних	26.11.2023 р.	
6	Проектування програмних засобів	03.12.2023 р.	
7	Захист роботи	14.12.2023 р.	

Студент  Дмитро ЧЕРНОВ
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  Матвій ІЛЬЯШЕНКО
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 77 с., 20 рис., 10 табл., 17 джерел.

ГРУНТ, МОДЕЛЬ, НЕЙРОМЕРЕЖІ, СИСТЕМА, СІЛЬСЬКОГОСПОДАРСЬКА ДІЯЛЬНІСТЬ

Метою магістерської роботи розробка моделі системи визначення найкращої культури для визначеної ділянки та розробка алгоритму визначення культури, що дозволяють підвищити ефективність сільськогосподарської діяльності.

Відповідно до поставленої мети виділено такі завдання роботи:

- провести аналіз предметної галузі, зокрема, існуючих способів визначення якості ґрунту;
- провести аналіз існуючих рішень для визначення найкращої культури для сільськогосподарської діяльності і їх розрахунків;
- визначити технології, які будуть необхідні для виконання роботи;
- розробити модель системи яка буде брати географічні дані ділянки та комбінувати дані з інтернету з введеними аналітичними даними для подальшого застосування в системі;
- натренувати модель яка б по заданим хімічним і погодним показникам визначала найбільш придатну культуру.

Об'єктом дослідження є процес визначення найкращої культури на певній сільськогосподарській ділянці.

Предметом дослідження є моделі та алгоритми розрахунку геохімічних показників якості ґрунту.

ABSTRACT

EN: 77 p., 20 figures, 10 tables, 17 sources.

AGRICULTURAL ACTIVITY, MODEL, NEURAL NETWORKS, SOIL, SYSTEM

The aim of the master's thesis is to develop a model of a system for determining the best crop for a particular area and to develop an algorithm for determining the crop, which will increase the efficiency of agricultural activities.

In accordance with this goal, the following tasks have been identified:

- analyze the subject area, in particular, existing methods of determining soil quality;
- to analyze existing solutions for determining the best crop for agricultural activities and their calculations;
- identify the technologies that will be required to perform the work;
- develop a model of the system that will take geographical data of the site and combine data from the Internet with the entered analytical data for further use in the system;
- to train the model to determine the most suitable crop according to the given chemical and weather indicators.

The object of the study is the process of determining the best crop on a particular agricultural plot.

The subject of the study is models and algorithms for calculating geochemical indicators of soil quality.

ЗМІСТ

Вступ.....	8
1 Аналіз способів визначення придатності земельної ділянки	11
1.1 Наземні методи визначення якості ґрунту.....	11
1.1.1 Датчик вологості ґрунту	12
1.1.2 Дистанційний та автоматизований експрес-контроль агрометеорологічних умов для рослинництва.....	13
1.2 Автоматизований та дистанційний експрес-контроль агрометеорологічних умов.....	14
1.3 Аерокосмічні методи визначення вологості і температури ґрунту	15
1.4 Агрогідрологічні властивості ґрунтів	16
1.5 Наявні рішення по визначенню якості ґрунту.....	18
1.6 Розрахунки якості ґрунту	20
1.6.1 Кислотність і лужність	21
1.6.2 Значення органічних речовин ґрунту та добрив для живлення рослин	23
1.6.3 Водний режим ґрунту	25
2 Вибір технологій для розробки системи.....	28
2.1 Вибір платформи для розробки	28
2.2 Технології для серверної частини.....	32
2.2.1 Java	33
2.2.2 Python.....	34
2.2.3 Node.js.....	34
2.2.4 PHP	36
2.2.5 Ruby	38
2.2.6 Підсумок.....	39
2.3 Варіанти баз даних	41

2.3.1 Продуктивність.....	41
2.3.2 Надійність	43
2.4 Вибір JavaScript фреймворку для клієнтської частини	45
2.4.1 Архітектура	45
2.4.2 Продуктивність.....	47
2.4.3 Висновки по вибору JavaScript фреймворку	50
3 Проєктування системи планування і аналізу для отримання максимального врожаю.....	51
3.1 Розробка засобів управління системою планування і аналізу для отримання максимального врожаю	51
3.2 Проєктування бази даних для системи	55
3.3 Логічна модель бази даних.....	61
3.4 Фізична модель бази даних	62
3.5 Тригери та збережені функції	63
3.6 Проєктування програмних засобів	65
3.7 Тестування на різних даних	69
Висновки	72
Перелік джерел посилання	74
Додаток А.....	76

ВСТУП

У сучасному світі питання продовольчої стійкості та ефективності сільського господарства виходить на перший план на тлі стрімкого зростання населення та викликів, пов'язаних зі зміною клімату. У міру зростання цих викликів аграрна галузь шукає інноваційні рішення для забезпечення продовольчої безпеки, сталого розвитку та ефективного використання земельних ресурсів. Одним з найважливіших рішень у сільському господарстві є вибір земельних ділянок для вирощування певних культур.

Ґрунт є майже невідновлюваним ресурсом, оскільки він відновлюється завдяки різноманітним довготривалим біологічним процесам вивітрювання гірських порід, які відбуваються в геологічних масштабах часу. Людська діяльність, така як урбанізація, вирубка лісів та індустріалізація, може спричинити ерозію та виснаження ґрунту. Для утворення нового ґрунту може знадобитися від сотень до тисяч років і більше, що робить його невідновлюваним ресурсом. Але комплексний підхід може відновити родючість ґрунту ¹

Історично процес вибору відповідної землі для вирощування був поєднанням мистецтва та науки. Фермери часто покладалися на знання поколінь, інтуїцію та локальні спостереження за станом ґрунту і кліматичними умовами. Хоча ці традиційні методи мають свої переваги і слугували людству протягом тисячоліть, їм іноді бракує точності та масштабованості, необхідних у сучасному складному сільськогосподарському ландшафті.

У таких країнах, як Україна, де сільське господарство відіграє ключову роль в економіці, ставки ще вищі. З її величезними орними землями Україну влучно називають "житницею Європи". Однак складність її різноманітних рельєфів означає, що те, що працює в одному регіоні, може бути непридатним в іншому. Наприклад, якість ґрунту може кардинально відрізнятися в межах

невеликих відстаней під впливом таких факторів, як попередні практики вирощування, доступність води та основні геологічні умови. Розуміння цих нюансів стає першочерговим для отримання оптимальної врожайності та сталого ведення сільського господарства.

Україна володіє надзвичайним резервом родючих ґрунтів. Тому в минулому вона була, і в майбутньому повинна бути одним із світових лідерів виробництва високоякісних, екологічно чистих продуктів харчування. Але для цього потрібна також враховувати й стан ґрунтів, їх родючість та інтенсивність забруднення хімічними елементами, залишками пестицидів, радіонуклідами, тощо. Необхідні також знання як впливають гідрометеорологічні умови на це забруднення.

Вологість і температура ґрунту - одні з головних фізичних властивостей ґрунтів, що визначають їх родючість. Без наявності необхідної кількості води в ґрунті і відповідної температури, сільськогосподарські культури не проростають зовсім. Вологість ґрунту впливає на розчинність, переміщення та ефективність органічних і мінеральних добрив, на ступінь забруднення ґрунту пестицидами та іншими продуктами техногенного походження, на те, скільки с.-г. рослини засвоюють шкідливих для здоров'я людей хімікатів.

Поточна інформація про вологість і температуру ґрунту в системі управління виробництвом с.-г. продукції необхідна, в першу чергу, для прийняття оперативних рішень при розробці екологічно чистих технологій вирощування сільськогосподарських культур з метою одержання максимального врожаю. Маються на увазі технологічні рішення (агротехнічні заходи) з питань основного і передпосівного обробітку ґрунту, доцільності і строків сівби (посадки) с.-г. культур, норм і строків внесення добрив, норм і строків поливу, тощо. Однак органи управління, наука, виробництво володіють далеко неповною оперативною інформацією про вологість ґрунтів через нестачу необхідних методів і технічних засобів їх одержання.

Більше того, проблеми не обмежуються лише розумінням землі. Сучасні фермери повинні враховувати й інші фактори, такі як мінливі вимоги ринку, еволюція сортів сільськогосподарських культур та мінливі кліматичні умови. Отже, процес прийняття рішень стає більш багатограним і вимагає системи, яка може асимілювати різні дані, щоб запропонувати найкращі рекомендації.

Саме тут технології, зокрема, досягнення в галузі штучного інтелекту та аналітики даних, пропонують безпрецедентний потенціал. Уявіть собі систему, яка може оцінити ділянку землі на основі історичних даних, поточного стану ґрунту, середніх погодних умов і навіть супутникових знімків, щоб передбачити її придатність для вирощування певної культури. Така система не лише зменшить кількість здогадок у сільському господарстві, але й максимізує врожайність та прибутковість, забезпечуючи при цьому стале використання землі.

Метою цієї дипломної роботи є заглиблення в розробку такої системи, пристосованої до українських ґрунтів та умов. Використовуючи можливості новітніх технологій, таких як нейронні мережі, системи управління базами даних та геоаналітика, дослідження має на меті революціонізувати традиційні методи вибору земельних ділянок для вирощування сільськогосподарських культур.

1 АНАЛІЗ СПОСОБІВ ВИЗНАЧЕННЯ ПРИДАТНОСТІ ЗЕМЕЛЬНОЇ ДІЛЯНКИ

1.1 Наземні методи визначення якості ґрунту

Для визначення вологості ґрунту в мережі агро- і гідрометеорологічних станцій України та в підрозділах інших відомств широко використовують метод повітряного або теплового висушування зразків ґрунту - термостатно-ваговий (ТВ) метод. Він також відомий як термогравіметричний метод. Найпростіший метод. За цим методом вологість ґрунту на сільськогосподарських угіддях визначають шляхом відбору зразків ґрунту з окремих шарів на глибині від 5 до 100-150 см за допомогою спеціального ґрунтового пробовідбірника (наприклад, АМ-26). Ці зразки поміщають в алюмінієві контейнери об'ємом близько 60 кубічних сантиметрів і транспортують до ґрунтової лабораторії, де визначають їхню вагу з вологим ґрунтом. Маса тих же контейнерів з сухим ґрунтом визначається після висушування в термостатах. За відомою масою порожнього контейнера вираховують вологість ґрунту. При правильному виконанні всіх перерахованих вище операцій цей метод дозволяє визначити вологість одного зразка ґрунту з досить високим ступенем точності, тому його часто використовують для калібрування вимірювачів вологості ґрунту.

Головними недоліками цього методу є:

- вологість визначається в одному зразку ґрунту, взятому з нижньої частини циліндра сівалки, а не в 10-сантиметровому шарі. Тому отримані дані носять випадковий характер і в більшості випадків не можуть бути розподілені по всьому полю.
- тривалість процесу визначення вологості ґрунту становить 1,5 - 2 дні, що спричиняє затримку у прийнятті технологічних рішень, адже у разі випадання значних опадів або суховіїв вологість ґрунту за цей період може суттєво змінитися.

- труднощі цього методу та вартість обладнання зумовлюють високу вартість отримання необхідної інформації про вологість ґрунту.

Серед інших методів і засобів, які можна спробувати використати для вимірювання вологості ґрунту слід відзначити стаціонарні датчики вологості ґрунту, надвисокочастотні вологоміри ґрунтових зразків, радіолокатори для визначення вологості ґрунту, фотооптичні вологоміри ґрунтових зразків, переносні ГЧ-вологоміри і ряд інших.

1.1.1 Датчик вологості ґрунту

Електричний датчик опору призначений для дистанційного моніторингу вологості ґрунту (рисунок 1). Основна стінка циліндра напівпроникна (пропускає лише воду). У центрі циліндра знаходяться два ізольовані електричні контакти та речовина, яка поглинає вологу з ґрунту. Якщо внутрішнє всмоктування датчика стає більшим, ніж всмоктування ґрунту, датчик буде поглинати воду і випускати воду, якщо всмоктування ґрунту стає більшим. Таким чином, осмос всередині датчика і осмос ґрунту завжди знаходяться в динамічній рівновазі. Діапазон осмотичних сил становить від 10 до 200 сантибарів. Вимірюваний електричний опір становить від 500 до 30 000 Ом. Недоліками цього типу датчиків є висока інерційність і неможливість використання на всіх полях, а тільки на тих, які не обробляються в даний момент (той же недолік, що і у нейтронного вологоміра). Крім того, датчик є сорбційним (абсорбційним), а це означає, що його сорбційна здатність може змінюватися з часом, а отже, і його калібрувальні параметри. Нарешті, ґрунт - це не інертне середовище, а активне. Активна, а іноді й агресивна природа цього середовища може призвести до того, що внутрішній наповнювач датчика почне хімічно взаємодіяти з ґрунтом, спричиняючи зміну його електричних параметрів калібрування.

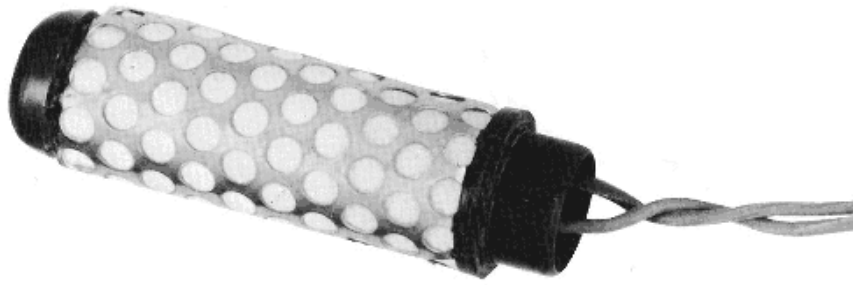


Рисунок 1.1 - Датчик вологості ґрунту

1.1.2 Дистанційний та автоматизований експрес-контроль агрометеорологічних умов для рослинництва

Призначений для швидкого визначення вологості ґрунту на гідрометеорологічних станціях та в сільському господарстві. Принцип його дії базується на вимірюванні ослаблення електромагнітної енергії мікрохвильового поля (НВЧ) вологими зразками ґрунту. За словами розробника, обраний метод вимірювання в діапазоні частот 10 ГГц дозволяє виключити вплив хімічного складу ґрунтів на результат вимірювання вологості, хоча це твердження потребує ретельної експериментальної перевірки. Основним і суттєвим недоліком є те, що він вимагає відбору зразків ґрунту за допомогою ґрунтового бура. Другим суттєвим недоліком приладу є його нерепрезентативність. Цей прилад і телевізійний метод не відображають просторові поля вологості, оскільки вологість ґрунту не визначається в 10-сантиметровому шарі всього поля, зайнятого певною культурою. Вона вимірюється і визначається тільки в зразку ґрунту (близько 40-50 грамів), взятому за допомогою ґрунтового бура. При цьому він вимірює вологість не на всьому полі, а в об'ємі ґрунту, схожому на форму циліндра, висотою 10 см і діаметром 60 см, що відповідає зразку вагою близько 40-50 кг сухого ґрунту, тобто набагато більшому, ніж 40-50-грамова проба. Фотооптичний вологомір для зразків ґрунту та портативний інфрачервоний вологомір IRMM-106 мають той самий недолік. Для того, щоб використовувати ці два прилади для вимірювання

вологості, необхідно брати дуже маленькі зразки з ґрунту за допомогою ґрунтового шнека.

1.2 Автоматизований та дистанційний експрес-контроль агрометеорологічних умов

На гідрометеорологічних станціях, в сільськогосподарських науково-дослідних інститутах, природоохоронних установах та на багатьох сільськогосподарських підприємствах виконується важка рутинна робота зі збору, обробки, зберігання та передачі інформації про стан навколишнього середовища, здоров'я рослин тощо. Ця робота вимагає матеріальних витрат і великої кількості ручної праці. Така робота пов'язана з матеріальними витратами і великою кількістю ручної праці, що призводить до значного збільшення собівартості інформації. Масштабна автоматизація цих робіт із застосуванням сучасних комп'ютерних технологій дозволила б не тільки зменшити кількість ручної праці, а й здешевити інформацію і, перш за все, значно підвищити її якість та оперативність. Розробка і застосування автоматизованих вимірювальних систем для збору, обробки, передачі та зберігання інформації про стан ґрунтів пов'язана з вирішенням низки важливих проблем, однією з найважливіших з яких є проблема датчиків вологості ґрунту.

Ні в Україні, ні в інших країнах не існує офіційно визнаного датчика для вимірювання вологості ґрунту, мабуть, найбільш критичного параметра. Але вся справа автоматизації вимірювання інших параметрів ґрунту в агрометеорології та агрономії є неефективною без вирішення проблеми датчиків вологості ґрунту. При наявності датчиків вологості ґрунту вже стають важливими температура, щільність, кислотність та інші параметри ґрунту, рослин і повітря. Створення та

впровадження дистанційних агрометеорологічних станцій може передувати широкій автоматизації вимірювань агрометеорологічних параметрів ґрунту. Такі станції дозволять добре протестувати датчики і вимірювальні системи, відпрацювати технологію вимірювань і, на початковому етапі, значно знизити трудомісткість і вартість первинної інформації про стан ґрунту. На більш пізньому етапі, без зміни інфраструктури мережі агрометеорологічних спостережень, ці станції можуть бути легко переведені в повністю автоматичний режим з додатковими матеріальними витратами.

1.3 Аерокосмічні методи визначення вологості і температури ґрунту

Аналіз патентів, науково-технічної літератури та інформації в Інтернеті щодо аерокосмічних методів моніторингу земної поверхні, зокрема рослинності та ґрунтів, свідчить про те, що наземним методам і засобам при розробці таких методів моніторингу приділяється недостатньо уваги. Моделі, що розробляються для оцінки стану рослинності та ґрунтів за супутниковими даними, які в основному базуються на законах оптики, термодинаміки та інших фізичних законах, завжди потребуватимуть експериментальної перевірки в наземних умовах. А для цього потрібна надійна наземна мережа моніторингу стану рослин і ґрунтів. Однак при цьому виникають дуже складні, але, тим не менш, важливі проблеми, пов'язані зі специфікою аерокосмічних і наземних методів.

Аерокосмічні методи не можуть бути використані для отримання розподілу температури і вологості ґрунту на глибинах до 1-1,5 м. У цьому випадку вони поки що не можуть замінити наземні засоби отримання інформації про вологість і температуру ґрунту.

Дані з супутників зазвичай не збігаються в часі та просторі з даними з землі. Траєкторія польоту супутників часто не збігається з розташуванням існуючої мережі гідрометеорологічних станцій і постів, які проводять наземні спостереження за станом атмосфери, рослин і ґрунту. Час проведення агрометеорологічних спостережень також часто не збігається з часом проходження штучних супутників над місцем розташування станцій і постів. Врахувати це можна двома способами:

- якщо мікрокліматичні характеристики території і добовий хід агрометеорологічних показників добре вивчені, необхідно внести відповідні поправки в дані наземних спостережень;
- другий спосіб. На шляху проходження штучного супутника слід заздалегідь створити мобільні агрометеорологічні пости з більш частою періодичністю спостережень, ніж у гідрометеорологічних станцій.

Такі пости не тільки дозволять більш точно ув'язати наземні спостереження з космічними даними, але й можуть слугувати новою, дуже ефективною технічною базою для мікрокліматичних досліджень території. Реалізація другого методу дозволить в майбутньому використовувати перший метод або комбінацію першого і другого методів.

1.4 Агрогідрологічні властивості ґрунтів

УкрНДГМІ підготував до друку посібник "Агрогідрологічні властивості ґрунтів України", в якому подано найповніший опис таких агрогідрологічних властивостей різних типів ґрунтів: об'ємної (щільності, об'ємної маси), вологості стійкого в'янення, непродуктивної вологи, мінімальної польової вологоємності та повної (максимальної) польової вологоємності. Крім того, наведено статистичні

дані про мінливість цих властивостей багатьох типів і різновидів ґрунтів, а також деякі практичні рекомендації щодо їх використання в агрометеорології та агромеліорації.

Дослідження показали, що використання даних з окремих ґрунтових розрізів для характеристики конкретного поля не завжди є виправданим з огляду на природну мінливість агрогідрологічних властивостей у часі та просторі. При об'єднанні груп даних, що відносяться до певних типів і різновидів ґрунтів, кожен таку групу даних можна розглядати як загальний набір. Мінливість таких агрогідрологічних властивостей, як щільність, мінімальна та максимальна польова вологоємність тощо, є такою ж, як і на одному гектарі поля. Це дає можливість екстраполювати середні значення агрогідрологічних властивостей ґрунту, отримані за відповідною методикою, навіть на поля, де такі спеціальні дослідження ніколи не проводилися.

На рисунку 1.2, наприклад, показано карту об'ємної маси ґрунтів України в шарі ґрунту 0-20. Аналогічні карти підготовлені для всіх шарів ґрунтів України до глибини одного метра. Вони були підготовлені з використанням комп'ютера і повинні слугувати основою для підготовки карт класифікаційних параметрів для різних типів ґрунтів у національному масштабі. Дослідження агрогідрологічних властивостей ґрунтів також дозволяють підготувати аналогічні карти по областях, районах та окремих господарствах, за умови наявності відповідних ґрунтових карт.

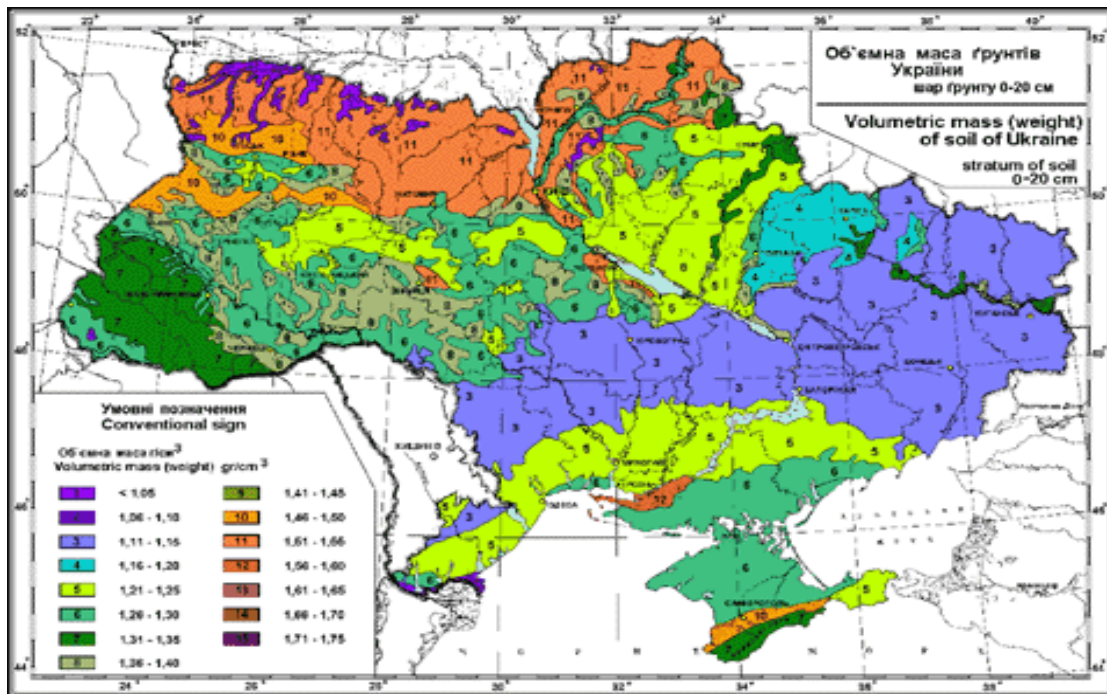


Рисунок 1.2 - Об'ємна маса ґрунтів України

1.5 Наявні рішення по визначенню якості ґрунту

Вибір правильного ґрунту для фермерського господарства є критично важливим кроком, який може створити або зруйнувати врожай. І хоча існує багато методів оцінки придатності ґрунту, доступ до них часто обмежений. Спеціалізовані дослідження, такі як аерофотозйомка, можуть надати цінну інформацію про стан ґрунту, але такі дані не завжди є специфічними для конкретної ділянки. Крім того, земля може переходити з рук в руки без належної передачі інформації про попередні агрохімічні аналізи, що ускладнює ідентифікацію потенційних інвесторів або фермерів.

Наразі в Україні є кілька компаній, які надають послуги з аналізу ґрунту на вміст пестицидів. Ці компанії працюють з великою кількістю зразків, збираючи ґрунт безпосередньо з полів, а потім проводять ретельні лабораторні дослідження

для оцінки таких параметрів, як родючість, рН, вміст поживних речовин та наявність токсичних елементів. На основі цих даних фермери отримують детальний звіт про стан своєї землі, який може допомогти їм спланувати внесення добрив та інші агрономічні заходи. Однак у послугах, які вони пропонують, є суттєва прогалина - вони не надають порівняльного аналізу між характеристиками ґрунту та конкретними потребами культур, які будуть вирощуватися.

Це означає, що навіть маючи повний хімічний склад ґрунту, фермери повинні розуміти, як ця інформація співвідноситься з потребами конкретних культур. Взаємозв'язок між хімічним складом ґрунту та потребами рослин має фундаментальне значення для забезпечення ґрунту необхідними поживними речовинами та підтримкою для здорового росту і розвитку. Якщо цей взаємозв'язок ігнорувати або неправильно інтерпретувати, можуть виникнути серйозні проблеми, починаючи від недостатньої врожайності і закінчуючи повною загибеллю врожаю. Ці проблеми неминуче призводять до економічних втрат для фермерів.

Таким чином, потреба в інноваційному інструменті, який міг би автоматизувати процес вибору земельних ділянок під посіви, стає очевидною. Такий інструмент має включати аналітику великих даних, штучний інтелект та інші сучасні технології, які б дозволили більш точно та науково обґрунтовано оцінити придатність ґрунту для різних видів культур, враховуючи їхні специфічні потреби.

Сучасний стан агротехнологічних рішень, на жаль, не повною мірою відповідає потребам сучасного аграрія, особливо в контексті українських аграрних реалій. Більшість наявних програмних рішень на ринку не інтегрують можливості штучного інтелекту, що обмежує їх здатність до адаптації під динамічні умови ведення сільського господарства, як-то кліматичні зміни, економічні коливання та специфічні потреби різних регіонів.

Хоча існують програмні рішення, які використовують GPS-навігацію для точного розмежування полів і автоматизації процесу збору зразків ґрунту, це лише початкові кроки в комплексному аналізі аграрних потреб. Станом на сьогодні, такі інструменти не включають в себе важливі фактори, як-от детальний аналіз погодних умов, економічну обґрунтованість посівів, чи актуальний стан ринку сільськогосподарської продукції, які є критично важливими для прийняття обґрунтованих рішень.

Оцінка погодних умов, наприклад, включає не лише короткостроковий прогноз, але й аналіз трендів і змін клімату в довгостроковій перспективі, що дозволяє планувати агротехнічні заходи з вищим ступенем точності. Фінансовий аналіз, з іншого боку, має враховувати не лише поточну вартість культур на ринку, але й прогнозовані зміни, спричинені економічними коливаннями, політичною ситуацією та іншими факторами.

У цьому контексті, розробка інтегрованої системи, яка би використовувала алгоритми штучного інтелекту для комплексного аналізу зазначених вище факторів, може стати справжнім проривом у сфері агротехнологій. Така система дозволила б аналізувати великі обсяги даних, від агрохімічного складу ґрунту до макроекономічних показників, забезпечуючи фермерам доступ до об'єктивної оцінки придатності ґрунтів під конкретні культури з урахуванням всіх ризиків.

1.6 Розрахунки якості ґрунту

При визначенні якості ґрунту для сільськогосподарських потреб, критично важливо розуміти, що кожна культура має унікальний набір вимог до умов зростання, який включає тип ґрунту, рівень рН, наявність поживних речовин, водозабезпечення, та багато інших факторів. Для прикладу, рис вимагає значної

кількості води і може рости в низько розташованих вологих умовах, тоді як виноград віддає перевагу дренованим, піщаним чи супіщаним ґрунтам з високим вмістом калію та вапняку.

Отже, для визначення якості ґрунту необхідно зібрати точні та об'єктивні дані, які відображають властивості ґрунту у відповідності до потреб певної культури. Це включає аналіз фізичних характеристик ґрунту, таких як текстура, структура, глибина профілю, здатність утримувати воду та повітря, а також хімічні показники, які охоплюють рівень рН, вміст органічних речовин, наявність макро- та мікроелементів.

Важливо не лише виміряти ці показники, а й інтерпретувати їх через призму специфічних вимог кожної культури. Розвиток інформаційних технологій та застосування алгоритмів штучного інтелекту можуть суттєво спростити цей процес, надаючи фермерам детальні та надійні рекомендації. За допомогою таких аналітичних інструментів можливо автоматично обробляти великі обсяги даних та визначати оптимальні умови для кожної культури, враховуючи не тільки нинішній стан ґрунту, а й потенційні зміни, що можуть виникнути в результаті кліматичних змін або інших зовнішніх впливів.

1.6.1 Кислотність і лужність

Ступінь кислотності або лужності, виражений в одиницях рН за шкалою теоретичних (можливих) значень від 1 до 14, є відносною кількістю іонів водню, H^+ , присутніх у ґрунті. Оскільки шкала є логарифмічною, зміна рН лише на одну одиницю означає десятикратну зміну кислотності або лужності. Наприклад, ґрунти з рН 5 у 10 разів кисліші, ніж ґрунти з рН 6, а ґрунти з рН 4 у 100 разів кисліші, ніж ґрунти з рН 6.

Ґрунти зі значенням рН 7 вважаються нейтральними, з меншим значенням - кислими, а з більшим - лужними. Ґрунти з рН 6,6 вважаються кислими в сільськогосподарському виробництві. Наприклад для пшениці найбільш сприятливими для росту і розвитку є значення рН від 5,5 до 7,5. Однак ці значення

pH можуть відрізнятися для різних ґрунтів, різних місць вирощування та різних сортів пшениці (Рисунок 1.3).

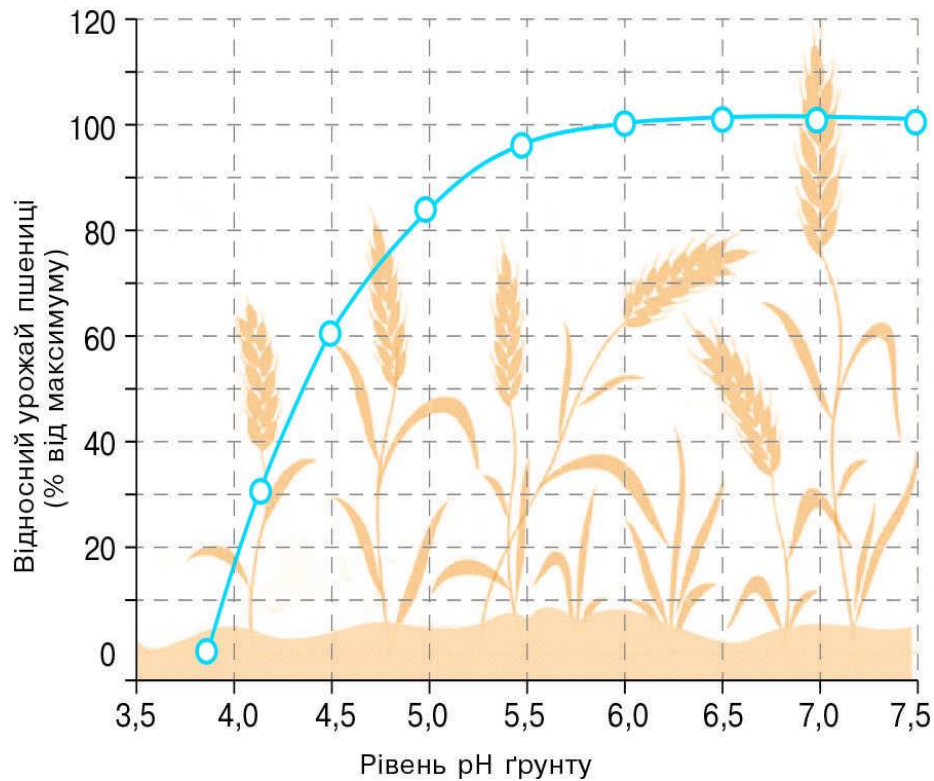


Рисунок 1.3 - Вплив pH ґрунту на врожай пшениці, виражений у відсотковому відношенні до можливого врожаю, якщо кислотність не буде обмежуючим чинником

Кислотність і лужність також впливають на багато біологічних процесів у ґрунті, в тому числі на розвиток патогенів рослин і бульбочкових бактерій, які розвиваються на коренях бобових рослин і здатні поглинати азот з атмосфери. Слід зазначити, що фіксація азоту бульбочковими бактеріями добре відбувається в нейтральних або лужних ґрунтах, тоді як у кислих ґрунтах цей процес гальмується.

Так, багато сільськогосподарських культур, які традиційно вирощують в Україні, віддають перевагу нейтральному або слабокислому ґрунтовому

середовищу. Оптимальний діапазон рН для більшості культур дійсно знаходиться між 4.5 та 7, з невеликими відмінностями залежно від конкретної рослини.

рН ґрунту впливає на розчинність поживних речовин, які є доступними для рослин, а також на активність мікробіологічних процесів у ґрунті. Коли рівень рН занадто низький (кислий), деякі поживні речовини стають недоступними, а інші – токсичними. Якщо ж рН занадто високий (лужний), це також може привести до нестачі певних мікроелементів, необхідних для росту рослин.

Оптимальний діапазон рівня рН ґрунту має вирішальне значення для успішного вирощування основних сільськогосподарських культур, які є традиційними для України. Для пшениці найкращим вважається рН в межах 6.0-7.0, яке забезпечує оптимальне засвоєння поживних речовин із ґрунту. Кукурудза віддає перевагу трохи ширшому діапазону рН, зазвичай від 5.5 до 7.5, що дозволяє виростити здорові і врожайні рослини навіть на дещо кислих ґрунтах. Соняшник також є досить адаптивним до різних рівнів рН, проте ідеальний діапазон для нього становить 6.0-7.5. Картопля, в свою чергу, вирощується краще всього у ґрунті з рН від 4.8 до 6.5, що робить його виборчим до кислішого середовища у порівнянні з іншими культурами.

1.6.2 Значення органічних речовин ґрунту та добрив для живлення рослин

Ґрунти містять величезну кількість поживних речовин у вигляді гумусу, органічних залишків рослинного походження та мінеральних сполук. Оціночні запаси гумусу, азоту та фосфору для різних ґрунтів є наступними (таблиця 1.1).

Азот ґрунту в основному зосереджений в гумусі, в середньому 5-6%, з невеликою часткою у вигляді мінеральних сполук. Значна частина фосфору також присутня в органічних сполуках. У підзолах, наприклад, органічний фосфор становить 20-25% від загальної кількості, а в чорноземах - близько 50%. Калій у ґрунтах знаходиться переважно в мінеральній формі. Валовий вміст калію у формі K_2 становить 60-80 тонн на гектар.

Таблиця 1.1 - Запаси гумусу, азоту та фосфору (в т / га) у різних ґрунтах

Ґрунти	Шар 0-20 см			Шар 0-100 см		
	хумус	Н	P ₂ O ₅	хумус	Н	P ₂ O ₅
Потужні чорноземи	160	11	2.7	700	36.0	11.9
Звичайні чорноземи	140	7	2.4	500	25.0	8.5
Підзоли	50	3.2	0.8	100	6.6	1.7
Сіроземи	40	3.7	0.7	80	7.5	1.4

Враховуючи, що для отримання врожаю пшениці у розмірі 20 ц/га в шарі чорнозему потрібно 40-60 кг азоту, 20-25 кг фосфору і 28-30 кг калію, то їх запаси в шарі чорнозему товщиною 20 см можуть забезпечити такий врожай на 170-230 років, а запаси підзолів і сірих ґрунтів можуть забезпечити такий врожай на 60-75 років. Метровий шар підзолів і сірих ґрунтів може забезпечувати високий вихід азоту протягом 130-150 років, такий же шар чорноземів - 500-700 років, а осушені землі - до 2500-2800 років. Мінеральні ґрунти мають запаси фосфору і калію на ще більшу кількість років. Певна кількість азоту, фосфору, калію та інших поживних речовин у ґрунті міститься в рослинних рештках (коріння, поживні рештки). Більшість цих залишків (до 250 т/га) міститься в цілинних ґрунтах. Хоча кількість поживних речовин у рослинних рештках відносно невелика, цінним є те, що ці речовини швидко перетворюються на сполуки, які можуть бути засвоєні рослинами.

Покращення перетворення гумусу та рослинних решток за допомогою належного обробітку ґрунту та сільськогосподарських практик є найважливішим фактором підвищення врожайності в країні. Однак найважливішим засобом збільшення валових зборів сільськогосподарської продукції, особливо в районах, де поширені підзолисті та опідзолені типи ґрунтів, є використання органічних

добрив, таких як гній, торф і різні види компостів. Їх використання часто подвоює і потроює врожайність сільськогосподарських культур.

1.6.3 Водний режим ґрунту

Хороші сходи пшениці та кукурудзи з'являються лише тоді, коли в шарі ґрунту 0 - 10 см міститься 10 мм і більше доступної вологи. Таким чином, у цій зоні для досягнення своєчасних сходів достатньо лише чорного пару та зрошення. У той же час, шар ґрунту 0 - 10 см часто містить лише 2-6 мм вологи або є повністю сухим після непарових попередників.

У таких випадках проростання залежить від опадів у післяпосівний період. Однак вони зазвичай випадають в середині жовтня або пізніше, що призводить до пізньої появи сходів, нестачі часу для укорінення до зими та поганого розвитку посівів.

Багаторічні дані показують, що якщо доступні запаси вологи в шарі ґрунту 0-20 см під час посіву становлять 20-30 мм, а в шарі 0-100 см - 100-115 мм і більше, рослини будуть добре вкорінені і перебуватимуть у доброму стані розвитку восени.

Рівень вологозабезпеченості рослин восени важливий не тільки для проростання озимої пшениці, але й для гарного розвитку рослин та отримання високих врожаїв. Існує тісний зв'язок між запасами вологи в метровому шарі ґрунту в період сівби та врожайністю пшениці ($r = 0,69-0,79$).

До кінця осінньої вегетації волога поповнюється за рахунок опадів, але незначно - в середньому 28 мм. У цей період запаси вологи в метровому шарі ґрунту на перелозі в середньому становлять 100-115 мм, а після перелогу - 60-80 мм. При цьому часто ґрунт зволожується лише на глибину 40-50 см, а глибші шари до 1 м залишаються майже сухими.

Найбільше вологи в посівах пшениці накопичується протягом осінньо-зимових місяців. Тому найбільші запаси ґрунтової вологи щорічно спостерігаються на початку весни (рисунки 1.4). У цей період в 1-метровому шарі

грунту середні запаси доступної води становлять 165 мм при зрошенні, 150 мм під паром і 130 мм після незрошеного пару.

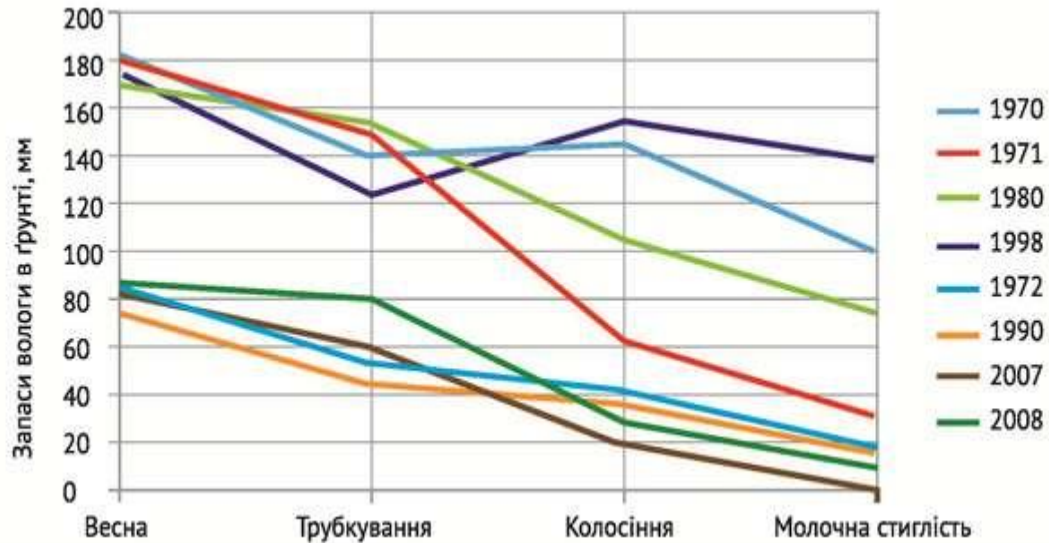


Рисунок 1.4 - Динаміка запасів доступної води в шарі ґрунту 0–100 см протягом вегетації пшениці озимої у роки з високими і низькими запасами води весною, мм

Навесні ґрунт вологий на глибину 1-1,5 метра, але бувають роки, коли він вологий лише на 50-60 см, а нижні шари напівсухі.

Кількість опадів, що випадають восени та взимку, достатня для накопичення до 170 мм доступної ґрунтової води. Насправді, дуже мало опадів поглинається ґрунтом у цей період. Так, із 196 мм опадів, що випали за осінньо-зимовий період, в середньому за 42 роки спостережень, в посівах пшениці після кукурудзи на силос, 83 мм води було поглинуто шаром ґрунту 0-100 см, 40 мм випаровуванням і 31 мм зрошенням. Таким чином, ґрунт поглинув лише 42, 20 і 16% опадів відповідно. Решта опадів - 58-84% - стікала в низини, частина випаровувалася або замерзала і втрачалася, не приносячи ніякої користі. Кількість

вологи з опадів, що поглинається посівами пшениці під паром і на зрошенні, є особливо низькою там, де ґрунт є більш вологим.

Дослідження показують, що за осінньо-зимовий період втрачається 100-160 мм вологи, чого достатньо для отримання 1-2 т зерна пшениці. Такі великі втрати є неприйнятними для посушливої зони, оскільки це, по суті, втрачений урожай.

Надзвичайно високі втрати вологи були виявлені під час шлюбного періоду з березня по вересень. Втрачається вся кількість опадів, що випала в шлюбний період - в середньому 232 мм - і ще 18 мм, або 14,4%, ґрунтової вологи, накопиченої раніше в осінньо-зимовий період. Сумарні втрати вологи за шлюбний період в середньому становлять 250 мм, або більше половини річної норми опадів.

Це пов'язано з тим, що інтенсивні літні опади на перелогових землях не встигають вбиратися і більша їх частина стікає і випаровується. Також відбувається значна втрата ґрунтової вологи під час глибокого літнього зябу, що є досить поширеним явищем.

Отже, при більшій кількості опадів на посівах пшениці та чорному пару водний режим ґрунту не покращився. Спостерігається навіть деяке від'ємне значення водного балансу, але це не є проявом аридизації території, оскільки ці відхилення незначні. Водночас, очевидно, що кількість втрачених опадів за останні роки значно зросла. Це свідчить про те, що ефективність використання опадів знизилася. Тому збільшення кількості опадів в останні роки майже не відчувається в сільському господарстві.

Це все є наслідком поточного способу ведення сільського господарства. Занадто розтратного використання землі і жорсткі методи обробки. Для вирішення цієї проблеми треба більше піклуватися про стан ґрунту, бо з часом врожаї ставатимуть меншими, тоді коли населення планети тільки збільшуватиметься.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Вибір платформи для розробки

На даний момент в арсеналі розробників програмного забезпечення існує широкий спектр варіантів для створення систем, кожен з яких має свої унікальні переваги та специфікації. Від веб-інтерфейсів, які забезпечують універсальний доступ через інтернет-браузери, до десктопних застосунків, що пропонують високу продуктивність та персоналізацію для кінцевих користувачів. Мобільні застосунки, з іншого боку, пропонують надзвичайну зручність та мобільність, дозволяючи користувачам отримувати доступ до інформації та управляти своїми даними на ходу. Навіть носимі гаджети, такі як смарт-годинники та фітнес-браслети, відкривають нові можливості для інтеграції технологій в повсякденне життя фермерів, надаючи їм оперативну інформацію прямо на зап'ясті.

При виборі платформи для розробки програмного забезпечення, важливо враховувати ряд ключових факторів. По-перше, зручність для користувача є критично важливою. Це означає, що інтерфейс має бути інтуїтивно зрозумілим, простим у використанні, і при цьому забезпечувати доступ до всіх необхідних функцій. По-друге, масштабованість системи є важливою, адже вона повинна мати можливість рости та адаптуватися до змінюваних потреб користувачів та ринкових умов. Також важливо, щоб система дозволяла легко вносити зміни та оновлення, що забезпечує її актуальність та безпеку в довгостроковій перспективі.

Правильний вибір платформи та технологій для розробки програмного забезпечення має велике значення, адже від цього залежить не лише задоволеність користувачів, але й ефективність, з якою програма може впроваджувати інновації в сільське господарство. Вибір між різними платформами залежить від цілей та потреб кінцевих користувачів, а також від специфіки сільськогосподарської діяльності, яку програма покликана покращити.

На основі попереднього досвіду роботи з веб-технологіями та враховуючи специфічні вимоги до системи, було прийнято рішення розробляти веб-сервіс. Основною перевагою такого підходу є здатність системи ефективно працювати з різноманітними даними в мережі Інтернет, що є особливо важливим в контексті аграрного сектору, де дані постійно змінюються та потребують актуалізації. Використання веб-сервісу також гарантує, що інформація буде зберігатися для подальшого використання та легко доступна для користувачів.

Для реалізації цього проекту було обрано JavaScript як основну мову програмування. JavaScript є універсальною мовою, що підходить для створення як клієнтських, так і серверних частин веб-додатків. Для серверної частини вибрано платформу Node.js, яка відома своєю високою продуктивністю та ефективністю у роботі з асинхронними операціями, що є критично важливим для веб-додатків, котрі обробляють великі обсяги даних. Nest.js, обраний як основний фреймворк, забезпечує структурованість коду та полегшує розробку за допомогою використання сучасних парадигм програмування, таких як Dependency Injection та модульність.

Для зберігання даних вибрано PostgreSQL - потужну систему управління базами даних, що підтримує складні запити та гарантує надійність і безпеку зберігання інформації. PostgreSQL також відома своєю високою масштабованістю та гнучкістю, що дозволяє легко адаптувати базу даних під зростаючі потреби проекту.

Для клієнтської частини вибрано React.js, популярну JavaScript-бібліотеку для побудови інтерфейсів користувача. React.js дозволяє створювати швидкі та відгуківі веб-інтерфейси, що забезпечують відмінний користувацький досвід. Використання фреймворку Next.js з React.js сприяє підвищенню продуктивності, оптимізації для пошукових систем та поліпшенню загальної структури проекту. Все це разом створює міцну технічну основу для розробки високоефективного та надійного веб-сервісу.

Хоча розробка мобільного застосунку також була розглянута як потенційний варіант, існує ряд обмежень, які зробили вибір на користь веб-сервісу більш виправданим. Розробка мобільного застосунку все одно буде потребувати створення бази даних та веб-інтерфейсів для обміну інформацією між застосунком і сервером, а також розробки модулів для виконання запитів в Інтернеті. Крім того, мобільні застосунки вимагають окремої розробки та підтримки для двох основних платформ: Android та iOS, що збільшує робочий обсяг і потребує додаткових ресурсів.

На відміну від мобільних застосунків, веб-сервіс надає значно більше контролю над середовищем виконання. Це дозволяє легше виявляти та виправляти помилки, а також забезпечує більшу гнучкість у впровадженні змін і оновлень. Веб-сервіси доступні з будь-якого пристрою, що має браузер і підключення до Інтернету, що робить їх надзвичайно зручними для користувачів, які можуть не мати специфічного мобільного пристрою або віддавати перевагу роботі з настільного комп'ютера чи ноутбука.

Крім того, веб-сервіси пропонують можливість використання технологій сервіс-воркерів, які дозволяють певну частину функціоналу застосунку працювати офлайн. Це може бути особливо корисним у сільському господарстві, де доступ до Інтернету може бути обмежений або недоступний у певних локаціях. Використання сервіс-воркерів дозволяє забезпечити доступ до важливих даних та функцій навіть у відсутності Інтернет-з'єднання, підвищуючи зручність та надійність системи для кінцевих користувачів.

Також не останнім фактором був попередній досвід з певними технологіями і практиками. Цей аспект також відігравав важливу роль у прийнятті рішення щодо вибору веб-сервісу як основної платформи для розробки. Наявність попереднього досвіду роботи з конкретними технологіями, такими як JavaScript, Node.js, React.js та інші, значно полегшила процес розробки. Глибоке розуміння

набуте в ході попередніх проектів, дозволило не лише прискорити процес розробки, але й забезпечити високу якість кінцевого продукту.

В ході розробки планувалося інтегрувати використання нейромереж, що є сучасним і перспективним напрямком у сфері обробки даних та штучного інтелекту. Нейромережі мали б виконувати критично важливі функції, такі як аналіз даних про стан ґрунту, прогнозування урожайності, ідентифікацію потенційних проблем з культурами та оптимізацію ресурсів на фермах. Використання таких технологій могло б значно підвищити ефективність та точність агрономічних рішень.

На відміну від настільних комп'ютерів, де існують добре розвинені технології для ефективного використання нейромереж, багато мобільних пристроїв мають обмеження щодо обчислювальної потужності та оптимізацій під цей тип завдань. Це стосується як апаратного забезпечення, так і програмного середовища. Наприклад, багато сучасних мобільних пристроїв не мають спеціалізованих процесорів або достатньої обчислювальної потужності для ефективного виконання важких нейронних мереж.

Ця обмеженість стала однією з причин, чому веб-сервіс був обраний як основна платформа. Використання веб-сервісу дозволяє централізувати обчислення та аналіз даних на потужних серверах, а не розподіляти їх між клієнтськими пристроями. Такий підхід забезпечує високу ефективність обробки даних, не залежить від обмежень клієнтських пристроїв і дозволяє використовувати складні нейромережі без компромісів у продуктивності. Крім того, це знижує вимоги до апаратного забезпечення кінцевих користувачів, роблячи систему доступнішою для широкого кола фермерів, незалежно від технічних можливостей їхніх пристроїв.

Враховуючи всі ці фактори, вибір розробки веб-сервісу, на відміну від мобільного застосунку, здається більш обґрунтованим та ефективним рішенням,

що дозволить надати користувачам гнучку, доступну, та високофункціональну систему.

2.2 Технології для серверної частини

Мови програмування Ruby, Python, PHP та Node.js відіграють ключову роль у бекенд-розробці. Їх ефективне використання для бекенд-розробки значною мірою залежить від правильного вибору та інтеграції з відповідними фреймворками, бібліотеками та середовищами виконання. Ефективне використання цих мов для бекенд-розробки залежить від правильного вибору та інтеграції з відповідними фреймворками, бібліотеками та середовищами виконання не тільки з точки зору особливостей самої мови, але й з точки зору того, як ці інструменти можуть бути використані разом для створення стабільної, безпечної та ефективної архітектури додатків.

Вибір правильних інструментів для конкретного проекту є ключем до успіху. Це включає в себе розуміння вимог до функціональності, продуктивності, масштабованості та безпеки. Сюди входить розуміння вимог до функціональності, продуктивності, масштабованості та безпеки, а також спільноти підтримки, наявності навчальних ресурсів і можливостей інтеграції з іншими системами та сервісами.

Крім того, правильне використання бібліотек і фреймворків забезпечує високий рівень повторного використання коду та ефективність розробки, дозволяючи командам зосередитися на унікальних аспектах своїх проєктів, а не витрачати час на розробку основних компонентів.

Ефективне використання середовищ виконання, які відповідають вимогам конкретного додатку, також має вирішальне значення. Це допомагає забезпечити

стабільність, швидкість і масштабованість програми, що особливо важливо в сучасному світі технологій, який швидко розвивається.

2.2.1 Java

Java, одна з найпопулярніших мов програмування у світі, використовується розробниками вже понад 20 років і відома своєю універсальністю. Її застосування варіюється від розробки додатків для смартфонів, зокрема на платформі Android, до роботи зі смарт-картами. Java широко використовується для створення настільних і корпоративних додатків, веб-сайтів, підключення до баз даних, обробки зображень, розробки графічних інтерфейсів і мережевих додатків.

Окрім своїх переваг, Java також має деякі недоліки. Для початківців Java може здатися менш дружньою до користувача, ніж такі мови програмування, як Python. Це пов'язано з більш суворою типізацією і необхідністю писати більше коду для реалізації тієї ж функціональності. Ці аспекти можуть ускладнити вивчення та розуміння мови для початківців.

Java також відома своїм відносно повільним часом виконання порівняно з деякими іншими мовами програмування. Це пов'язано з тим, що програми на Java працюють на віртуальній машині Java (JVM), яка додає рівень абстракції між програмою та апаратним забезпеченням. Хоча це забезпечує переносимість коду, це також може призвести до збільшення використання пам'яті та погіршення продуктивності.

Інша проблема полягає в тому, що Java вимагає регулярних оновлень для підтримки безпеки та ефективності, що може ускладнити супровід коду, особливо для великих проектів і систем. Крім того, незважаючи на свою популярність і широке використання, Java може бути складно розширювати та інтегрувати з новітніми технологіями в деяких сценаріях розробки.

Ці фактори роблять Java важливим вибором для багатьох розробників, але її використання вимагає обережності та врахування специфіки проекту.

2.2.2 Python

Відомий своєю легкістю у вивченні, читабельністю коду та гнучкістю, Python є однією з найпопулярніших мов програмування сьогодні. Вона використовується в широкому спектрі додатків, від веб-розробки до наукових досліджень, від аналізу даних до штучного інтелекту та машинного навчання.

Але поряд з перевагами, Python має і деякі недоліки. По-перше, його часто критикують за відносно низьку продуктивність. Як інтерпретована мова, Python може працювати повільніше, ніж скомпільовані мови, такі як C або Java. Це особливо актуально, коли потрібно виконати інтенсивні обчислення або обробити великі обсяги даних.

Крім того, Python часто покладається на сторонні бібліотеки та фреймворки для реалізації певної функціональності, що може призвести до виникнення залежностей та проблем сумісності. Це може ускладнити управління проектом, особливо якщо бібліотеки часто оновлюються або мають конфлікти версій.

Ще одним недоліком є те, що Python не завжди є найкращим вибором для мобільної розробки. Хоча існують інструменти для створення мобільних додатків на Python, вони часто не можуть конкурувати з мовами та фреймворками, розробленими спеціально для мобільної розробки, такими як Swift для iOS або Kotlin для Android.

У випадках, коли потрібна висока продуктивність у реальному часі або тісна інтеграція з апаратним рівнем, наприклад, у вбудованих системах або розробці ігор, Python також може мати певні обмеження.

Незважаючи на ці недоліки, читабельність, гнучкість і широкий спектр застосувань Python означають, що вона залишається дуже популярною і широко використовуваною.

2.2.3 Node.js

JavaScript, який почав свій шлях як мова для створення динамічних веб-сторінок у браузері, з роками трансформувався в одну з найпотужніших та

найбільш універсальних мов програмування. Ця мова здобула особливу популярність та визнання завдяки розвитку платформи Node.js, яка дозволила використовувати JavaScript для розробки на сервері, розширюючи його можливості та області застосування.

Основні Переваги JavaScript та Node.js:

- універсальність: Однією з ключових переваг JavaScript є його універсальність. Завдяки Node.js, JavaScript може бути використаний як на клієнтській (у браузері), так і на серверній стороні, що дозволяє розробникам використовувати одну і ту ж мову програмування для всієї платформи. Це спрощує процес розробки та підтримки проектів.

- велика екосистема: JavaScript має одну з найбільших екосистем, завдяки величезній кількості доступних пакетів у npm (Node Package Manager). Це забезпечує розробникам швидкий доступ до готових рішень та бібліотек для різних завдань.

- висока продуктивність: Node.js використовує неблокуючий ввід/вивід та асинхронне програмування, що забезпечує високу продуктивність, особливо в додатках, які виконують багато операцій з вводом-виводом або обробляють великі обсяги даних.

- гнучкість у розробці: JavaScript дозволяє гнучко підходити до розробки, від простих скриптів до складних додатків, завдяки своїй динамічній природі та багатству функціональності.

Обмеження JavaScript та Node.js:

- управління залежностями: Через велику кількість залежностей, які можуть бути в проектах на Node.js, управління цими залежностями та забезпечення їх безпеки може стати викликом.

- асинхронне програмування: Хоча асинхронність є перевагою з точки зору продуктивності, вона також може створювати складнощі в розумінні та управлінні потоком виконання програми, особливо для новачків у JavaScript.

- проблеми з продуктивністю для деяких типів завдань:** Для деяких типів обчислювально інтенсивних завдань JavaScript може не бути найефективнішим вибором порівняно з іншими мовами програмування, які оптимізовані для таких завдань.

Загалом, JavaScript разом з Node.js є дуже потужною комбінацією для розробки сучасних веб-додатків, від простих веб-сайтів до складних інтерактивних додатків. Його універсальність, велика спільнота та широкі можливості роблять його фаворитом особисто для мене.

2.2.4 PHP

Як одна з основних мов веб-розробки, PHP має низку значних переваг, але водночас має й певні недоліки, які необхідно враховувати.

З одного боку, PHP відома своєю чудовою інтеграцією з багатьма серверами і системами управління базами даних, а також з іншими технологіями веб-розробки, такими як HTML, CSS і JavaScript. Це робить її ідеальним вибором для створення веб-сайтів. Гнучкість PHP також означає, що вона підтримує широкий спектр стилів програмування і може бути налаштована для виконання різних завдань, що робить її ідеальною для широкого спектру проектів. Ще одним важливим плюсом, що сприяє популярності PHP, є наявність великої спільноти та ресурсів для навчання і підтримки. Крім того, PHP є основою для багатьох популярних систем управління контентом.

З іншого боку, існують певні проблеми, які можуть обмежити використання PHP в деяких сценаріях. По-перше, це продуктивність: для дуже високонавантажених або обчислювально інтенсивних додатків PHP може бути не найкращим вибором. PHP також часто асоціюється з проблемами безпеки. Його популярність і відкритість роблять його частою мішенню для кібератак, хоча сучасні версії PHP зробили значні поліпшення в цій області. Іншим важливим фактором є те, що PHP іноді сприймається як застаріла мова розробниками, які шукають більш сучасні підходи та технології. Нарешті, у великих або складних

системах, де важлива висока масштабованість та ефективне управління ресурсами, певні архітектурні обмеження PHP можуть стати бар'єром для її використання.

Отже, щоб підсумувати про PHP:

- легка інтеграція та сумісність: PHP легко інтегрується з різними веб-серверами та базами даних. Ця мова програмування є основою багатьох динамічних веб-сайтів і підтримується більшістю хостинг-провайдерів, що робить її доступною для широкого використання;
- гнучкість у розробці: PHP дозволяє розробникам швидко створювати і редагувати скрипти, роблячи процес розробки більш гнучким. Це особливо корисно у ситуаціях, коли потрібно швидко внести зміни або оновлення;
- велика спільнота та ресурси: PHP має велику та активну спільноту, що надає велику кількість навчальних матеріалів, форумів для обговорення та документації. Це робить PHP доступнішим для новачків та надає підтримку досвідченим розробникам;
- питання продуктивності та оптимізації: Хоча PHP достатньо швидкий для багатьох веб-додатків, його продуктивність може страждати у високонавантажених або складних обчислювальних сценаріях. Це може вимагати додаткової оптимізації коду та конфігурації сервера;
- безпека: Хоча останні версії PHP включають значні поліпшення у безпеці, PHP-додатки часто потребують ретельного підходу до кодування та налаштування для запобігання загрозам безпеки, таким як SQL-ін'єкції та XSS-атаки;
- сприйняття у спільноті: На фоні нових, більш сучасних технологій, PHP іноді сприймається як застаріла мова, хоча вона продовжує розвиватися та адаптуватися;
- складність в Управлінні Великими Додатками: В управлінні великими, складними проектами, PHP може мати певні обмеження, особливо коли мова йде

про масштабованість та архітектурну складність. Розробка великих систем вимагає ретельного планування та використання передових практик кодування.

Загалом, PHP продовжує залишатися однією з найпопулярніших мов для веб-розробки завдяки своїй доступності, багатій функціональності та широкій підтримці, хоча і має певні обмеження. Це робить його ідеальним вибором для багатьох розробників, особливо для створення веб-сайтів та додатків, де необхідна швидка розробка та гнучкість.

2.2.5 Ruby

Ruby, як мова програмування, відома своєю виразністю та ефективністю, і має ряд як переваг, так і обмежень, які впливають на її використання в розробці.

Серед переваг Ruby, особливо важливо виділити його лаконічний та чистий синтаксис, що робить код легко читабельним і зрозумілим. Це значно полегшує процес навчання та входження в мову для новачків, а також сприяє швидкому розвитку проектів. Ruby також славиться своєю гнучкістю, дозволяючи розробникам легко змінювати внутрішню структуру мови під свої потреби, що робить її особливо привабливою для тих, хто хоче експериментувати або впроваджувати нетрадиційні рішення.

Крім того, Ruby має потужний фреймворк для розробки веб-додатків — Ruby on Rails, який значно спрощує процес розробки веб-додатків, надаючи стандартизований набір засобів та бібліотек. Цей фреймворк славиться своєю швидкістю розробки, що дуже цінно в умовах обмежених термінів та ресурсів.

Щодо обмежень, одним з найбільш очевидних є продуктивність. Ruby може бути повільнішим у порівнянні з деякими іншими мовами, особливо коли йдеться про обробку великих обсягів даних або виконання складних обчислювальних задач. Це може становити проблему для високонавантажених додатків, де продуктивність є критичною.

Іншим обмеженням є те, що, незважаючи на велику популярність Ruby on Rails, сам Ruby не є таким популярним у більш широких сферах програмування,

як інші мови, такі як JavaScript або Python. Це може обмежувати використання Ruby в деяких типах проектів або вимагати від розробників володіння додатковими мовами для роботи над більш різноманітними проектами.

В цілому, Ruby залишається важливим інструментом у веб-розробці, особливо завдяки Ruby on Rails, пропонуючи зручність, швидкість розробки та високу гнучкість. Хоча мова має свої обмеження, вона все ще вважається однією з найкращих варіантів для розробки сучасних веб-додатків, особливо для тих, хто цінує елегантність коду та швидкість розробки.

2.2.6 Підсумок

Взявши загальновідомі факти про найпоширеніші технології і порівнявши їх один з одним маємо наступну картину. На рисунку 2.1 видно, який стан ринку станом на 2021 рік. Під час аналізу та порівняння цих мов я зробив висновок, що вибір мови програмування для конкретного проекту має базуватися на кількох критичних факторах.

Перший і, можливо, один з найважливіших аспектів – це швидкість розробки. Деякі мови, як Ruby з Ruby on Rails, надають інструменти для дуже швидкої розробки додатків, що є ідеальним у випадках, коли час є критичним фактором. Інші мови можуть зажадати більше часу на реалізацію тих же функціональних можливостей, але водночас забезпечити більшу гнучкість або продуктивність.

Другий фактор, який варто враховувати, – це кількість доступних бібліотек і фреймворків. Наприклад, JavaScript з Node.js має величезну екосистему пакетів, що дозволяє швидко додавати нову функціональність. Така різноманітність може значно спростити процес розробки та розширити можливості проекту.

Третім важливим аспектом є легкість прочитання коду та підтримки. Деякі мови, як Python, славляться своїм чистим та легко читабельним кодом, що полегшує роботу у великих командах та спрощує процеси підтримки та оновлення проектів.

Останнім, але не менш важливим, є доступність навчальних матеріалів та гайдів. Для мов, як PHP та JavaScript, існує безліч ресурсів для навчання, що робить їх привабливими для новачків або для тих, хто хоче швидко розібратися з новою технологією.

Всі ці фактори в сукупності формують основу для вибору відповідної мови програмування та технологій для конкретного проекту, виходячи з його унікальних вимог та цілей. Важливо зважати на всі ці аспекти, аби забезпечити ефективну та успішну розробку.

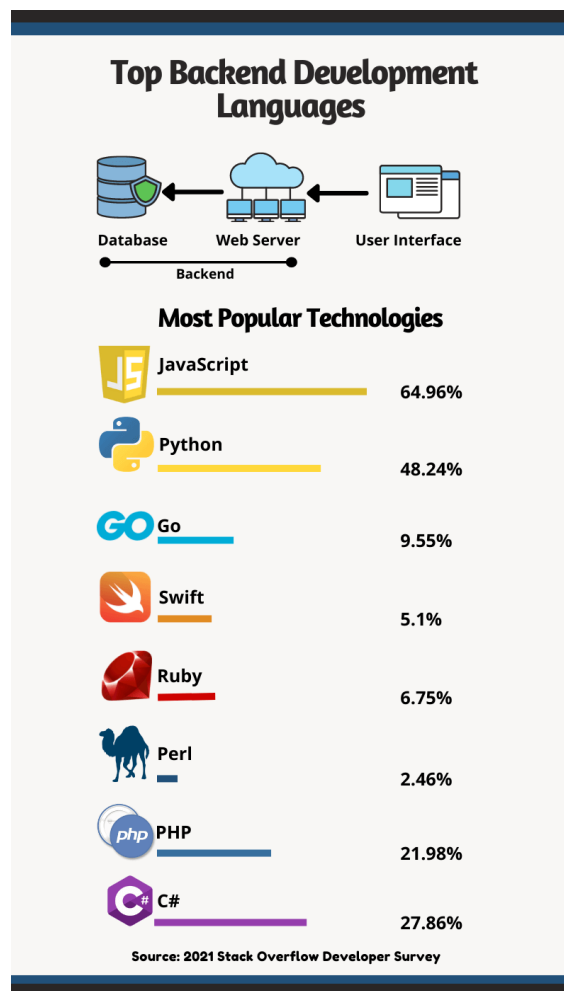


Рисунок 2.1 - Частка технологій в секторі веб-розробки станом на 2021 рік

Взявши до уваги всі фактори і факти було вирішено вести розробку бекенд частини системи мовою JavaScript з платформою Node.js. В якості фреймворка був обраний Nest.js як найпопулярніший і перевірений в жорстких умовах корпоративної розробки фреймворк.

2.3 Варіанти баз даних

Вибір бази даних для дипломної роботи вимагає розгляду декількох ключових факторів. Ці фактори включають в себе продуктивність, масштабованість, надійність, підтримку складних запитів і аналітичних функцій, а також легкість інтеграції з іншими технологіями, як-от Node.js.

2.3.1 Продуктивність

Розглянемо детальніше аспект продуктивності для кожної з баз даних: PostgreSQL, MySQL, MongoDB та Microsoft SQL Server. Продуктивність бази даних - це її здатність ефективно обробляти запити та транзакції, особливо у контексті великих обсягів даних та складних операцій.

PostgreSQL:

- оптимізація запитів: PostgreSQL має вдосконалений планувальник запитів, який ефективно оптимізує складні запити, забезпечуючи високу продуктивність;
- обробка великих даних: Він добре справляється з великими обсягами даних завдяки ефективному використанню індексів і зберігання даних;
- паралельна обробка: Сучасні версії PostgreSQL підтримують паралельну обробку запитів, що значно покращує продуктивність при роботі з складними запитами та аналітикою.

MySQL:

- використання індексів: MySQL також ефективно використовує індекси, особливо у випадку простих та середньо складних запитів;
- Оптимізація для читання: Загалом, MySQL оптимізована для операцій читання, що робить її добрим вибором для веб-додатків з високим рівнем читання даних;
- Обмеження в складних запитах: Для дуже складних запитів, особливо тих, що включають багато таблиць та агрегацій, MySQL може показувати меншу продуктивність порівняно з PostgreSQL.

MongoDB:

- оптимізація для документно-орієнтованих даних: MongoDB як NoSQL база даних ефективно обробляє великі обсяги неструктурованих або напівструктурованих даних;
- швидкість запису та читання: Вона відома своєю високою швидкістю запису та читання, особливо в розподілених середовищах;
- горизонтальна масштабованість: MongoDB ефективно масштабується горизонтально, що підходить для додатків з великим рівнем трафіку та обсягів даних.

Microsoft SQL Server:

- інтеграція з .NET: Microsoft SQL Server має відмінну інтеграцію з .NET, що може підвищувати продуктивність у середовищах, що базуються на Microsoft;
- паралельна обробка: Подібно до PostgreSQL, він підтримує паралельну обробку, що покращує продуктивність при роботі з великими обсягами даних;
- ефективність для складних запитів: SQL Server добре справляється зі складними транзакціями та запитами, пропонуючи різні інструменти для оптимізації продуктивності.

Кожна з цих баз даних має свої сильні сторони з точки зору продуктивності. Вибір між ними повинен базуватися на конкретних вимогах

проекту, таких як тип даних (структуровані проти неструктурованих), складність запитів, потреба в швидкому читанні або запису, та вимоги до масштабованості.

2.3.2 Надійність

Надійність баз даних може бути розглянута через лінзу кількох ключових аспектів: транзакційна цілісність, захист даних, відмовостійкість та відновлення після збоїв.

PostgreSQL:

- забезпечує строгу ACID-сумісність, що є фундаментальним для забезпечення надійності, особливо в застосуваннях, де цілісність даних є критичною;
- відома своєю стабільністю та надійністю у довгостроковій перспективі;
- містить вбудовані механізми для резервного копіювання та відновлення, що забезпечує додатковий шар захисту.

MySQL:

- надійність MySQL значною мірою залежить від вибраного движка зберігання, наприклад, InnoDB, який підтримує ACID;
- пропонує різні опції для резервного копіювання та відновлення, що дозволяє ефективно реагувати на втрати даних.

MongoDB:

- як NoSQL база даних, MongoDB була спочатку розроблена з акцентом на продуктивність та масштабованість, але останні оновлення внесли значні поліпшення в ACID-транзакції;
- ефективна в управлінні розподіленими даними, пропонуючи реплікацію та відмовостійкість;
- система підтримує гнучкі механізми резервного копіювання та відновлення, адаптовані для роботи з великими та розподіленими даними.

Microsoft SQL Server:

- як продукт, орієнтований на корпоративні потреби, SQL Server пропонує розширені можливості для забезпечення цілісності та безпеки даних;
- забезпечує повну підтримку ACID-транзакцій, що є важливим для бізнес-додатків з високими вимогами до надійності.

Надійність кожної бази даних залежить від конкретних сценаріїв використання та вимог до системи. PostgreSQL видається як міцна опція для застосувань, що вимагають високої цілісності та стабільності. MySQL є гнучким вибором для різноманітних веб-додатків. MongoDB вирізняється в обробці великих та розподілених наборів даних. Microsoft SQL Server є оптимальним для корпоративних систем з високими вимогами до безпеки та надійності.

Для фінального порівняння я оберу двох фаворитів - MySQL та PostgreSQL (таблиця 2.1).

Таблиця 2.1 - Підтримка функціональностей СКБД MySQL та PostgreSQL

Функціональність	MySQL	PostgreSQL
Відповідність вимогам ACID	Більшість двигунів підтримують ACID, але MyISAM не підтримує ACID	Повністю
Бекапи та Відновлення	Є деякі функції для бекапів для відновлення	Добре відомий своїми функціями по бекапам та відновленню
Підтримка декількох платформ	Так	Оптимізовано під UNIX системи
SQL типи даних	Стандартні типи	Стандартні + додаткові типи
Збережені процедури	Підтримуються	Більш просунуті з PLSQL
Тригери	Так	Так, гнучкі, різними мовами

І отже на основі порівняння двох СКБД я вважаю, що для цієї роботи буде за найкраще використати PostgreSQL.

2.4 Вибір JavaScript фреймворку для клієнтської частини

З розвитком веб-розробки 3 фреймворки JavaScript стали добре відомі всім фронтенд-розробникам: React, Vue.js та Angular.

React вважається бібліотекою інтерфейсу користувача, Angular - повноцінним фронтенд-фреймворком, який надає власні інструменти для всіх функцій, пов'язаних з розробкою веб-додатків, а Vue - просунутий фреймворк, реалізований як додаткова розмітка для HTML.

Для створення компонентних фронтенд-додатків з розширеними можливостями користувацького інтерфейсу всі три фреймворки можна використовувати майже взаємозамінно. Однак остаточний вибір залежить від вимог проекту та уподобань розробника.

2.4.1 Архітектура

По перше розберемо React. Ви можете почати використовувати React, написавши лише кілька рядків коду, і він не вимагає спеціальної структури проекту.

React-елементи - це основні будівельні блоки React-додатків. Вони більш потужні, ніж стандартні DOM-елементи. Це тому, що React DOM гарантує, що вони ефективно оновлюються з кожною зміною. Більші будівельні блоки, які називаються компонентами, визначають незалежні, багаторазові частини, які використовуються в усьому додатку. Вони приймають вхідні дані, які називаються пропсами, і утворюють елементи, які потім представляються користувачеві. Найпоширеніший спосіб структурувати проект - розбити інтерфейс на компоненти, наприклад, імпортувати його в основний компонент App.js, а потім відрендерити в React DOM.

Як і React, Vue.js представляє собою бібліотеку яка сконцентрована над шаром View в розрізі моделі MVVM. Vue не пов'язаний з паттерном Model-View-

ViewModel (MVVM). Однак, його дизайн частково натхненний ним. У Vue ви будете працювати переважно на рівні ViewModel, щоб гарантувати, що дані програми обробляються таким чином, щоб фреймворк відображав реальний вигляд і стан програми з мінімальною кількістю рендерингів.

Шаблонний синтаксис Vue дозволяє створювати компоненти, використовуючи комбінацію звичного HTML та спеціальних директив і функцій. Цей синтаксис шаблонів є кращим, хоча чистий JavaScript і JSX також підтримуються. Компоненти у Vue невеликі та самодостатні. Їх можна повторно використовувати у всьому додатку. Унікальною особливістю Vue є компонент Single File Component (SFC), який має розширення .vue. Він складається з трьох частин - шаблону, скрипту і стилю - значення яких інтуїтивно зрозуміле будь-якому розробнику.

На відміну від попередніх двох, Angular - це фреймворк MVC (Model-View-Controller). Але Angular 2 не має строгого зв'язку з патернами MV, оскільки він також базується на компонентах.

Модулі, компоненти та сервіси використовуються для структурування проектів в Angular. У кожному Angular-додатку є принаймні один кореневий компонент і один кореневий модуль.

Angular-компоненти - це контейнери для узгодженого блоку коду для прикладної області, робочого процесу або тісно пов'язаного набору функцій. Вони можуть містити компоненти, постачальників послуг та інші файли коду, обсяг яких визначається модулями, які вони містять. Вони можуть імпортувати функціональність, яка була експортована іншими модулями, або навпаки.

Кожен компонент Angular містить шаблон, логіку з метаданими і, за бажанням, стилі, розділені в окремих файлах, на відміну від Vue. Кожен компонент містить шаблон, логіку і, за бажанням, стилі, розділені в окремих файлах, на відміну від Vue. Шаблони Angular написані на мові HTML, але також

можуть включати синтаксис шаблону Angular зі спеціальними директивами, як у Vue.

Сервіси в Angular використовуються компонентами для делегування завдань, таких як отримання даних або перевірка вхідних даних. Вони є окремою частиною Angular-додатків, на відміну від React і Vue, які вимагають встановлення додаткових бібліотек для досягнення подібних цілей. Хоча Angular не змушує вас використовувати їх, ми наполегливо рекомендуємо структурувати додатки як набір окремих багаторазових сервісів.

Angular побудований на основі TypeScript, тому рекомендується використовувати саме його, хоча звичайний JavaScript також підтримується.

2.4.2 Продуктивність

Іншим фактором є продуктивність фреймворку або бібліотеки. У багатьох випадках продуктивність не є проблемою, особливо якщо ви працюєте над невеликим проектом. Однак, коли проект зростає і стає більш складним, продуктивність стає проблемою.

Розглянемо те, як наведені фреймворки показують себе у тесті від Google Chrome Labs:

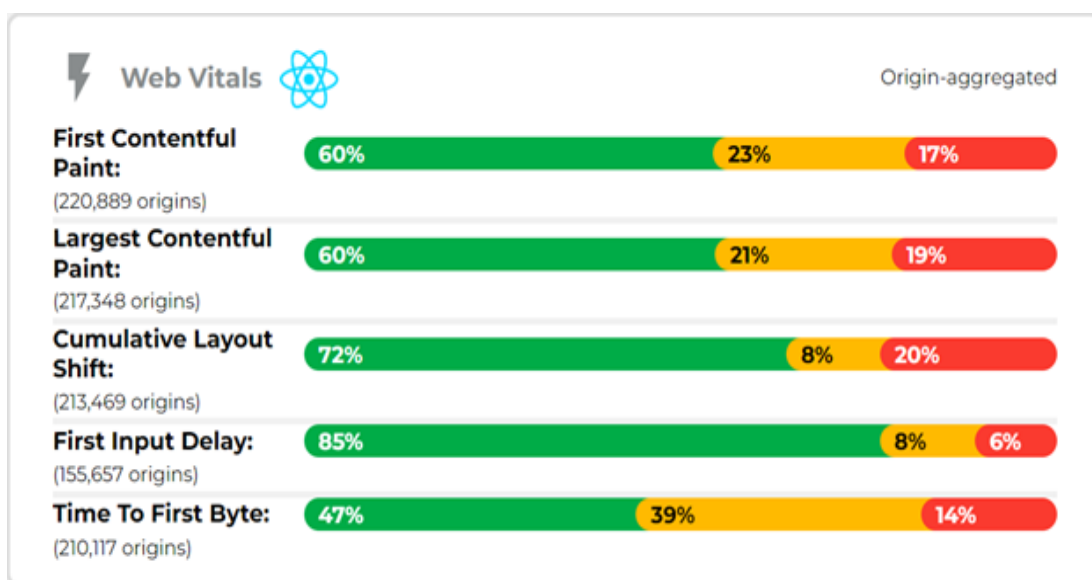


Рисунок 2.2 - Продуктивність React застосунка

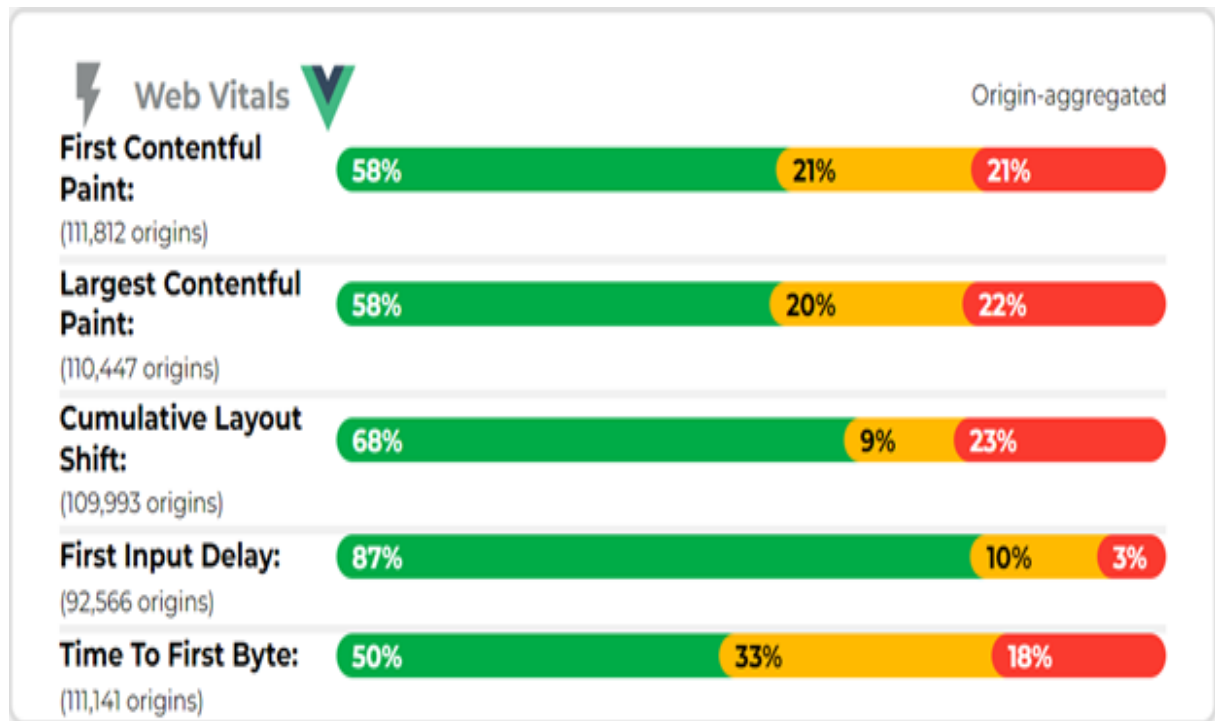


Рисунок 2.3 - Продуктивність Vue застосунка

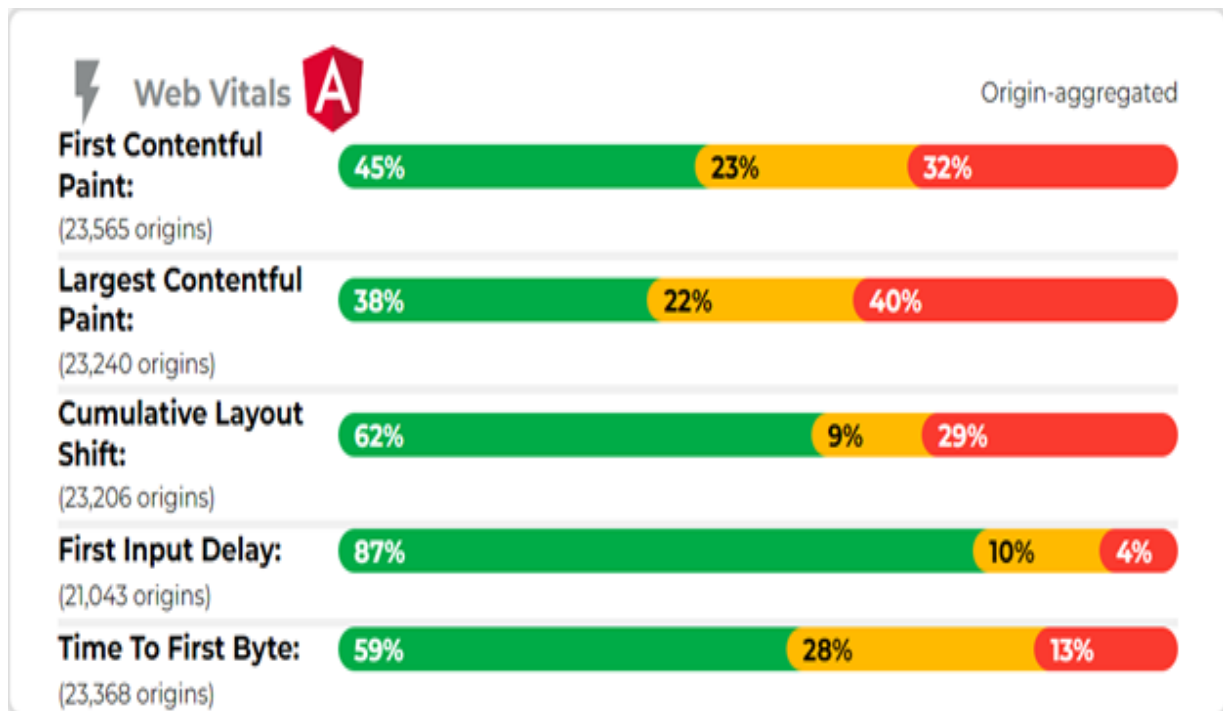


Рисунок 2.4 - Продуктивність Angular застосунка

В аналізі продуктивності фреймворків React (рисунок 2.2), Vue (рисунок 2.3) та Angular (рисунок 2.4) важливо враховувати різні аспекти їхнього рендерингу та взаємодії з користувачем. Час завантаження та рендерингу сторінки є одним з ключових показників, які впливають на загальний досвід користувача і продуктивність веб-додатків.

Vue і React, як зазначається, зазвичай завантажують і рендерять сторінку швидше, ніж Angular. Це може бути пов'язано з більш легковажною архітектурою і меншим обсягом попередньої обробки, яку виконують ці фреймворки перед відображенням контенту. Vue та React ефективно оптимізують процес рендерингу, мінімізуючи час, який необхідний для відображення першого контенту та для найбільшого блоку контенту на сторінці.

First Input Delay (FID), який вимірює час між дією користувача (наприклад, натисканням кнопки) та відповіддю сторінки, є важливим показником відгуку інтерфейсу. За цим параметром усі три фреймворки показують хороші результати, що свідчить про їх здатність швидко реагувати на взаємодію користувачів.

Особливо варто відзначити, що за даними досліджень, Vue частіше використовується у проектах, де загальний обсяг JavaScript коду не перевищує 1 МБ. Це свідчить про високу ефективність і легковаговість Vue, що робить його ідеальним вибором для невеликих та середніх проектів, де критично важливі швидкість завантаження та продуктивність. З іншого боку, Angular, який часто вимагає більшого обсягу пам'яті для своїх додатків, зазвичай краще підходить для більших, більш складних проектів, де потрібна розширена функціональність та більші можливості інтеграції.

Ці спостереження підкреслюють важливість вибору відповідного фреймворку на основі конкретних потреб проекту, розміру і складності додатку та очікувань щодо продуктивності та взаємодії з користувачем.

2.4.3 Висновки по вибору JavaScript фреймворку

На основі аналізу та порівняльного вивчення різних фреймворків для розробки клієнтської частини веб-додатків, мій вибір схиляється на користь React. Це рішення обумовлене кількома ключовими факторами, які стосуються як продуктивності, так і ефективності розробки:

- React використовує віртуальний DOM, який значно підвищує швидкість рендерингу у порівнянні з традиційним DOM. Це мінімізує кількість взаємодій з реальним DOM, які є витратними з точки зору продуктивності, і дозволяє створювати високопродуктивні інтерфейси користувача;
- пропонує розробникам компонентний підхід, що дозволяє побудувати структуровані, легко підтримувані додатки. Реюзабельні компоненти спрощують процес розробки та забезпечують більшу консистентність коду;
- широкий функціонал і бібліотеки: однією з ключових переваг React є наявність великої кількості відкритих бібліотек та інструментів. Це включає в себе стейт-менеджери (наприклад, Redux), маршрутизатори (React Router) та бібліотеки для роботи з API (наприклад, Axios). Такий багатий набір інструментів розширює можливості розробника та спрощує вирішення складних задач;
- React має велику та активну спільноту розробників, що є важливим фактором для будь-якого технологічного стеку. Велика кількість навчальних матеріалів, форумів та документації робить процес навчання та вирішення потенційних проблем значно простішим;
- легко інтегруватися з іншими бібліотеками та фреймворками, як на стороні клієнта, так і на сервері. Це дозволяє розробникам використовувати React у різноманітних проектах, поєднуючи його з різними технологіями та підходами.

Отже, обираючи React, можна очікувати не тільки високу продуктивність застосунків, але й ефективний та гнучкий процес розробки, завдяки широкому спектру інструментів та ресурсів, доступних для розробників.

3 ПРОЄКТУВАННЯ СИСТЕМИ ПЛАНУВАННЯ І АНАЛІЗУ ДЛЯ ОТРИМАННЯ МАКСИМАЛЬНОГО ВРОЖАЮ

3.1 Розробка засобів управління системою планування і аналізу для отримання максимального врожаю

Проведений аналіз показав, що така система має видавати на виході відсоток придатності обраної ділянки для подальшої коректної експлуатації та збору стабільно високого врожаю. Система повинна забезпечувати функціональність для двох ролей - адміністратора та фермера. Нижче, (в таблиці 3.1) наведено більш детальний опис для цих ролей.

Таблиця 3.1 - Опис ролей в системі

Назва	Опис
Адміністратор	Керує користувачами системи, створює нових, коригує їх дані. Здійснює операції над проаналізованими ділянками. Відновлює доступ до системи.
Фермер	Створює запити на аналіз ділянки. Вносить дані, необхідні для більш точного аналізу.

Отже в системі буде дві ролі. Основна задача фермера - це створити коректний запит, заповнити необхідні поля, окреслити ділянку в програмі. Адміністратор же буде займатись виключно підтримкою користувачів, надавати доступ та менеджети дані.

Основний функціонал системи буде складатись з декількох комплексних дій:

- окреслення ділянки на мапі для визначення географічних даних, отримання деталей по погоді і загальної характеристики регіону з системи інтернет;
- внесення даних після аналізу ґрунту. Так як ґрунт змінює свої характеристики з року в рік - аналізи необхідно брати регулярно. Можна використовувати мапи і діаграми ґрунтів по регіонам, але для більш точних результатів - необхідно вносити більше змінних.

Для ефективного виконання задач, пов'язаних із збором та аналізом даних земельної ділянки, створення відповідного інтерфейсу користувача є ключовим. Такий інтерфейс повинен бути інтуїтивно зрозумілим та ефективним, дозволяючи користувачам легко вводити, переглядати та аналізувати геометричні дані та геохімічні показники ділянки. Це може включати функціонал для введення параметрів, таких як розмір ділянки, тип ґрунту, рівень рН, рівень вологості та інші геохімічні характеристики. Окрім цього, інтерфейс має бути оснащений засобами для візуалізації цих даних, що допоможе користувачам краще розуміти та інтерпретувати інформацію, необхідну для прийняття обґрунтованих рішень щодо використання та обробки земельної ділянки.

У веб-інтерфейсі, розробленому на основі React.js та Next.js, користувачі - у нашому випадку фермери - мають змогу вручну вводити дані про розташування та геометричні характеристики своїх земельних ділянок. Цей процес здійснюється за допомогою інтеграції з сервісом карт Here maps, який вирізняється своїм широким функціоналом та великим безкоштовним лімітом на використання API. Таке використання сервісу Here maps дозволяє користувачам точно відобразити розташування та контури своїх ділянок на карті.

Фермери можуть створювати полігони на карті для визначення меж своїх ділянок. Важливою особливістю системи є те, що вона дозволяє створювати полігони будь-якої форми та розміру, не накладаючи обмежень на формування кордонів. Це надає користувачам гнучкість у визначенні точних геометричних

характеристик своїх земельних ділянок, що є критично важливим для точного аналізу та планування землекористування.

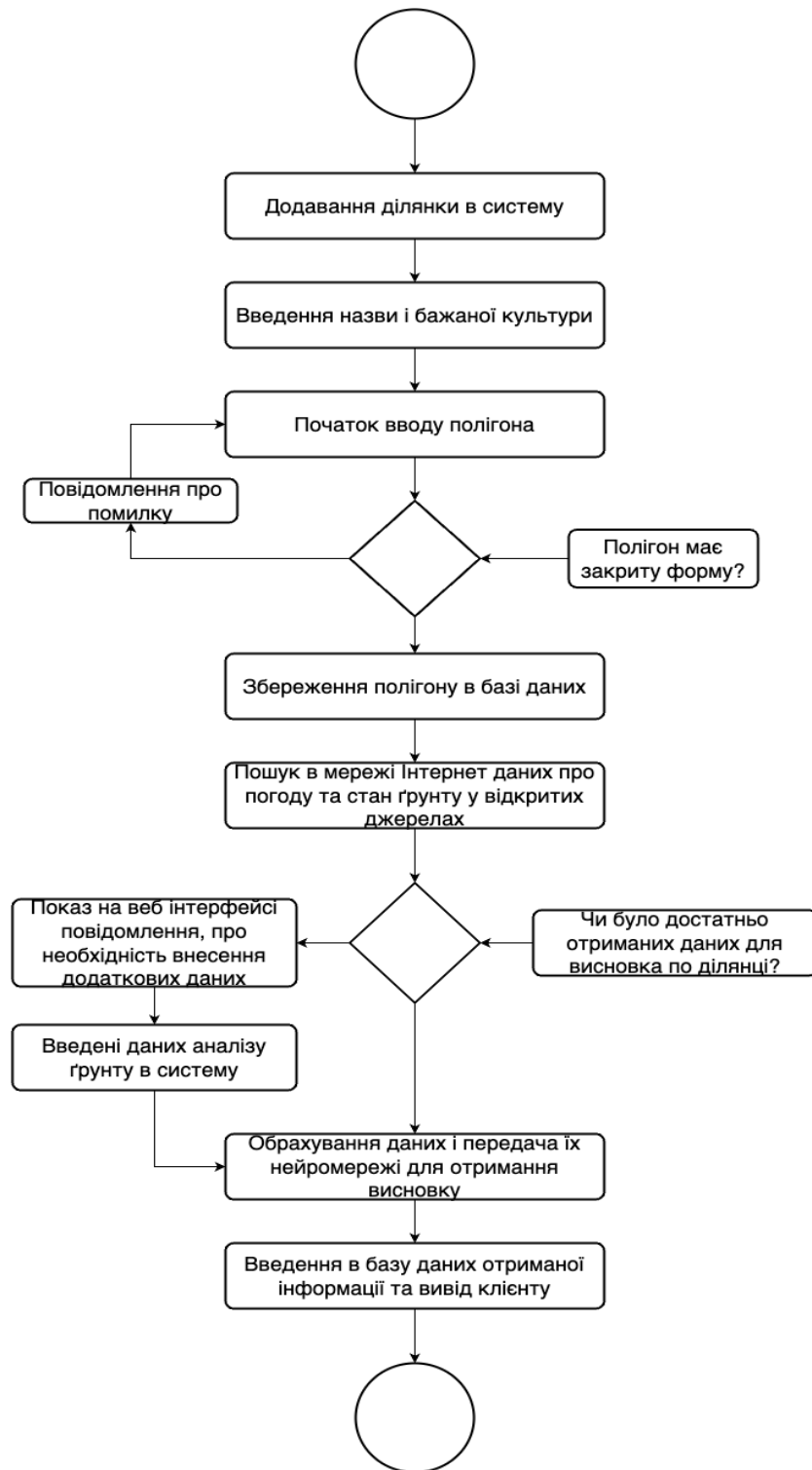


Рисунок 3.1 - Діаграма діяльності для додавання ділянки в систему

Однак, для отримання найбільш точних та коректних результатів аналізу, важливо, щоб фермери вводили адекватні та точні дані про свої ділянки. Некоректне визначення меж може призвести до неточностей у подальших розрахунках та аналізах, які здійснюються на основі цих даних. Тому інтерфейс також може включати настанови або підказки для користувачів щодо введення даних, щоб забезпечити максимальну точність та корисність введеної інформації.

На рисунку 3.1 зображено дії, які необхідно виконати фермеру для отримання результату

Нижче наведений рисунок 3.2, який показує діаграму діяльності для показу вже введених в систему ділянок.

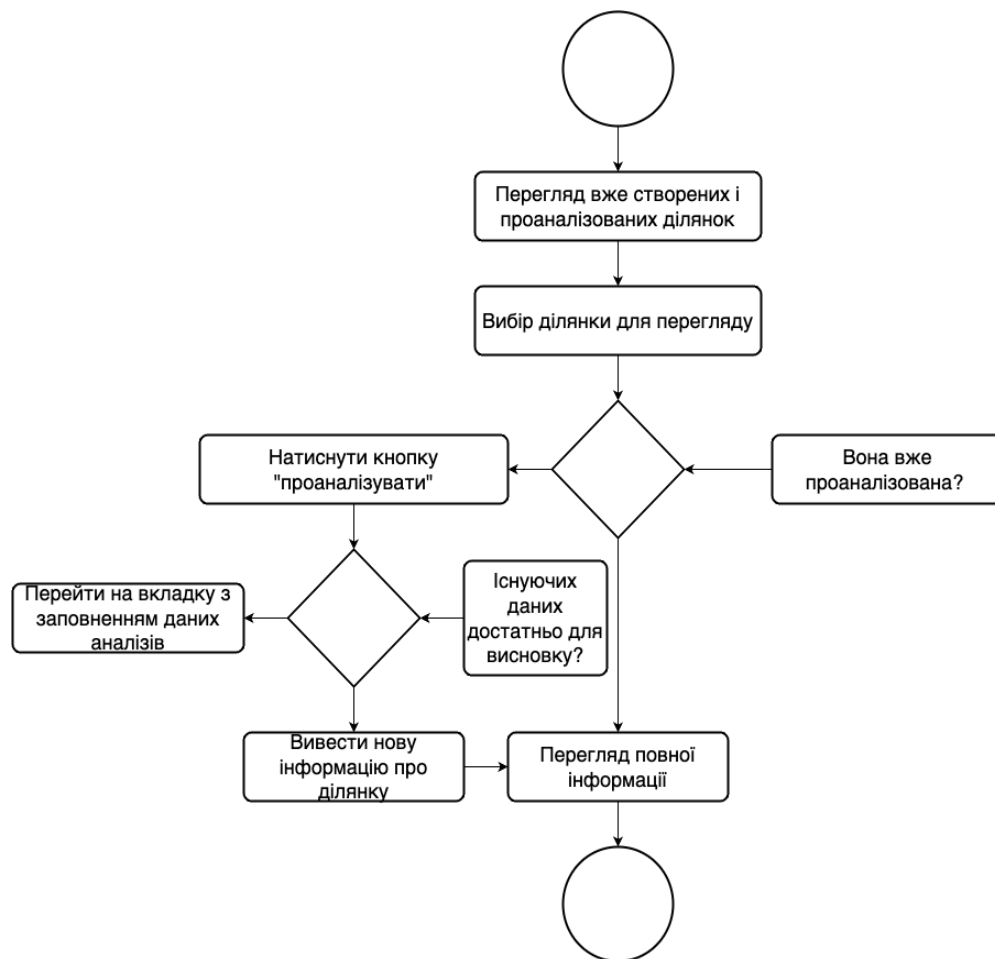


Рисунок 3.2 - Діаграма діяльності при перегляді існуючої сутності ділянки в системі

3.2 Проектування бази даних для системи

У процесі проектування бази даних для нашої системи одним з ключових аспектів є можливість створення і зберігання повторних записів одних і тих самих земельних ділянок з різних років. Ця особливість дозволяє фермерам не лише відслідковувати поточний стан своїх ділянок, але й аналізувати їх урожайність та придатність для певних культур у динаміці. Такий підхід відкриває можливості для глибокого аналізу змін у якості ґрунту, впливу агротехнічних заходів та змін кліматичних умов на урожайність.

У структурі бази даних кожна ділянка буде мати унікальний ідентифікатор, але зможе бути асоційована з декількома записами, які представляють різні роки. Для кожного року будуть зберігатися ключові показники, такі як тип ґрунту, рівень рН, вологість, показники родючості, а також дані про вирощувані культури та отриманий урожай. Це дозволить системі зіставляти дані різних років, виявляти тренди та зміни в стані ґрунту і урожайності.

Також важливою є можливість візуалізації цих даних у зручному для користувача форматі. Це може включати таблиці, графіки змін у часі, карти розподілу певних характеристик по ділянці. Така функціональність не тільки полегшить аналіз даних, але й допоможе у прийнятті рішень щодо подальшої стратегії обробітку земель і вибору культур.

Створення бази даних відбувається у кілька етапів:

- побудова концептуальної моделі - побудова семантичної моделі предметної області, тобто інформаційної моделі на найвищому рівні абстракції. Створення такої моделі не пов'язане з конкретною СКБД та моделлю даних. Терміни "семантична модель", "концептуальна модель" та "інформаційна модель" є синонімами. На цьому етапі відбувається визначення об'єктів, зв'язків між об'єктами, атрибутів і ключових атрибутів;

- логічне проектування - створення схеми бази даних на основі певної моделі даних, наприклад, реляційної моделі бази даних. Для реляційної моделі даних це набір реляційних діаграм, зазвичай із зазначенням первинних ключів і "зв'язків" між відношеннями, які є зовнішніми ключами. Перетворення концептуальної моделі на логічну зазвичай відбувається за формальними правилами. Цей етап можна значною мірою автоматизувати. Етап логічного проектування враховує специфіку конкретної моделі даних, але може не враховувати специфіку конкретної системи управління базами даних;

- фізичне проектування - це створення схеми бази даних для конкретної СКБД. Обмеження на іменування об'єктів бази даних, обмеження на підтримувані типи даних та інші можуть бути включені в специфіку конкретної СКБД. Фізичне проектування також включає в себе вибір фізичного середовища зберігання даних (вибір методів управління дисками, розбиття бази даних на файли і машини, методи доступу до даних), створення індексів і так далі.

Список сутностей наведено нижче у таблиці 3.2

Таблиця 3.2 - Список сутностей у базі даних системи

Назва	Опис
Адміністратор	Сутність містить інформацію про адміністраторів
Фермер	Сутність містить інформацію про фермерів
Ділянка	Сутність містить зберігає дані про ділянку
Аналіз	Сутність зберігає геохімічні показники ділянки
Висновок	Сутність зберігає висновки на основі проведених аналізів

Стовпці і їх опис наведені у таблицях нижче. Так як база даних використовується PostgreSQL - то існують тип даних під uuid. Тип даних text в цій базі даних буде залежати від вмісту і буде зберігати строкові дані до 65535 байтів.

Таблиця 3.3 містить стовпці сутності “Адміністратор”. Адміністратор має суто функціональні можливості і містить дані про логін, пароль, час створення. Сутність не прив’язана до інших таблиць.

Таблиця 3.3 - Сутність “Адміністратор”

Ім’я стовпця	Опис	Тип даних	Розмірність	Можливість значення NULL	Роль	Приклад
uuid	Унікальний ідентифікатор	uuid	128 bit	ні	PK unique	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
login	Логін в системі для ідентифікації та авторизації	text		ні		admin@gmail.com
password	Пароль, необхідний для авторизації в зашифрованому вигляді	text		ні		1b21hb2hb1u2gu3g2fxy1v2ux1v2y3vu12g4u3ggvgu43598sa89da98sd79adshuavusdva9sdguasd
created_at	Час створення запису	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00
updated_at	Час оновлення запису	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00

Таблиця 3.4 містить стовпці і їх опис для сутності “Фермер”. Це поля необхідні для авторизації, ідентифікації, наприклад логін, пароль, ім’я, прізвище, фотографія. Також ця сутність прив’язана до таблиць ділянки через зовнішній ключ.

Таблиця 3.4 - сутність “Фермер”

Ім'я стовпця	Опис	Тип даних	Розмірність	Можливість значення NULL	Роль	Приклад
uuid	Унікальний ідентифікатор	uuid	128 bit	ні	PK unique	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
email	Емейл в системі для ідентифікації та авторизації	text		ні		farmer@company.com
password	Пароль, зашифрований	text		ні		1b21hb2hb1u2gu3g2fxy1v2ux1v2y3vu12g4u3
name	Ім'я	text		так		
surname	Прізвище	text		так		
image_url	Картинка для аватару	text		так		
created_at	Час створення запису	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00
updated_at	Час оновлення запису	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00
deleted_at	Час видалення з системи	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00

Сутність “Ділянка” розписана в наступній таблиці 3.5. Тут будуть зберігатись інформація про назву ділянки, полігон з її розмірами, посилання на таблицю фермера та інформаційні поля.

Таблиця 3.5 - сутність “Ділянка”

Ім'я стовпця	Опис	Тип даних	Розмірність	Можливість значення NULL	Роль	Приклад
uuid	Унікальний ідентифікатор	uuid	128 bit	ні	PK unique	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
name	Назва ділянки в системі	text		ні		admin@gmail.com
polygon	https://postgis.net/docs/ST_GeometryType.html	geometry	як у текста			010200000000200000000000000000000000000000000000000000000f03f0
desired_crop	Бажана культура	text		ні		apple
farmer_uuid	Посилання на таблицю фермера	uuid	128 bit	ні	FK references uuid	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
created_at	Час створення запису	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00
updated_at	Час оновлення	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00
deleted_at	Час видалення	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00

Сутність аналізу це основа інформаційної частини про ділянку. Там будуть зберігатись дані температури, вологості, вміст азоту, фосфору, калію, кислотності і опадів.

Таблиця 3.6 - сутність “Аналіз”

Ім'я стовпця	Опис	Тип даних	Розмірність	Можливість значення NULL	Роль	Приклад
uuid	Унікальний ідентифікатор	uuid	128 bit	ні	PK unique	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
nitrogen	азот	real	4 bytes	так		58.0
phosphorus	фосфор	real	4 bytes	так		44.0
potassium	калій	real	4 bytes	так		40.0
farmplot_uuid	Посилання на таблицю ділянки	uuid	8 bytes	ні	FK references uuid	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
temperature	Температура	real	4 bytes	так		20.87
humidity	Відносна вологість	real	4 bytes	так		82.15
rainfall	Кількість опадів в мм	real	4 bytes	так		202.93
ph	Кислотність ґрунту	real	4 bytes	так		5.71
created_at	Час створення	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00

Дані про висновок після аналізу зберігаються в таблиці, і розписані в таблиці 3.7. Вони включають в себе ідентифікатор, час створення, назву культури для якої проводився тест і оцінка придатності ґрунту для вирощування, також є прив'язка до сутності “Аналіз”.

Таблиця 3.7 - сутність “Висновок”

Ім'я стовпця	Опис	Тип даних	Розмірність	Можливість значення NULL	Роль	Приклад
uuid	Унікальний ідентифікатор	uuid	128 bit	ні	PK unique	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
crop_name	Назва культури	text		ні		apple
analysis_uuid	Посилання на таблицю аналізу	uuid	8 bytes	ні	FK references uuid	a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
created_at	Час створення	timestamp	8 bytes	ні		1999-01-08 04:05:06 -8:00

З наявних даних можна побудувати логічну і фізичну модель бази даних, які представлені далі як наочних схем.

3.3 Логічна модель бази даних

Логічні моделі - це моделі бази даних, що розробляється і не обмежені якоюсь конкретною СКБД. Створення логічної моделі - це визначення конкретних сутностей, атрибутів і зв'язків між ними. На цьому етапі розробки бази даних можна призначити атрибутам типи даних. Етап проектування логічної моделі також дозволяє визначити первинні та зовнішні ключі, які будуть пов'язані з іншими сутностями.

На додаток до позначення сутностей і атрибутів, також треба вказати зв'язки між сутностями. Зв'язки "один-до-багатьох" і "один-до-одного" є

основними типами зв'язків у цій моделі. Наприклад, зв'язок "один-до-багатьох" присутній між сутностями "Ділянка" і "Аналіз", та "один-до-одного" між сутностями "Аналіз" і "Висновок", оскільки на кожний аналіз буде окремий висновок по с-г культурам.

На рисунку 3.3 зображена схема логічної моделі бази даних, що розробляється.

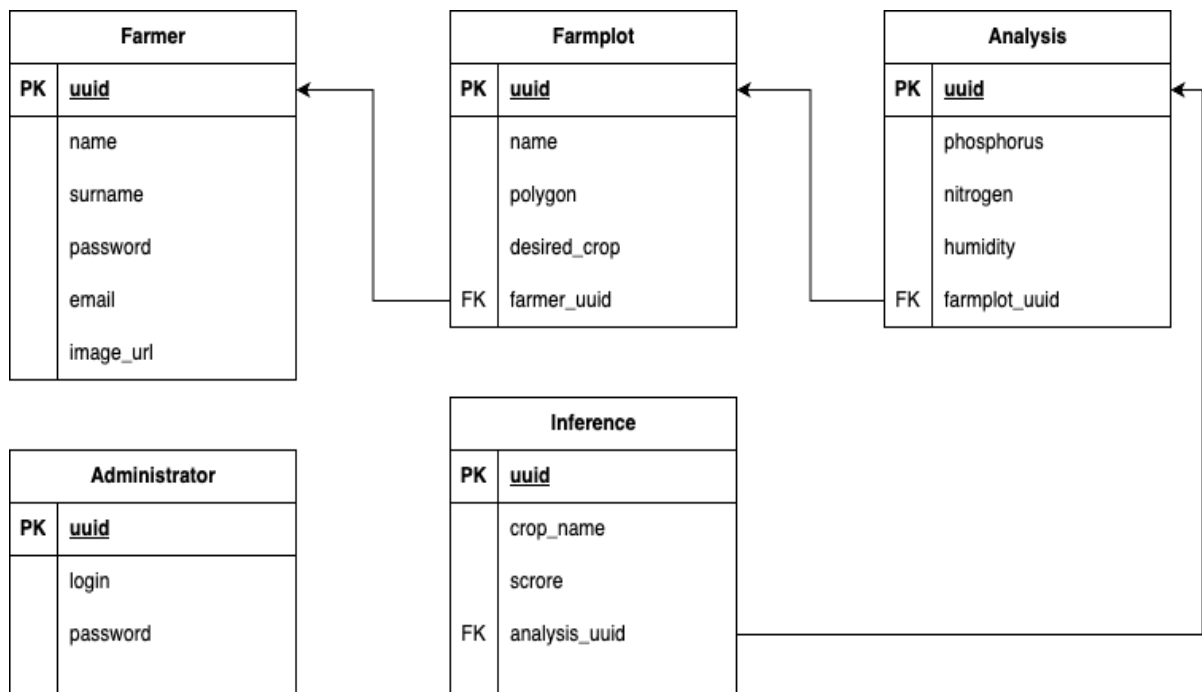


Рисунок 3.3 - логічна модель бази даних

Для наступного етапу побудуємо фізичну модель бази даних.

3.4 Фізична модель бази даних

Фізичною моделлю часто називають реалізовану логічну модель з використанням конкретної системи керування базами даних (СКБД).

На рисунку 3.4 зображено фізичну модель бази даних проектованої системи.

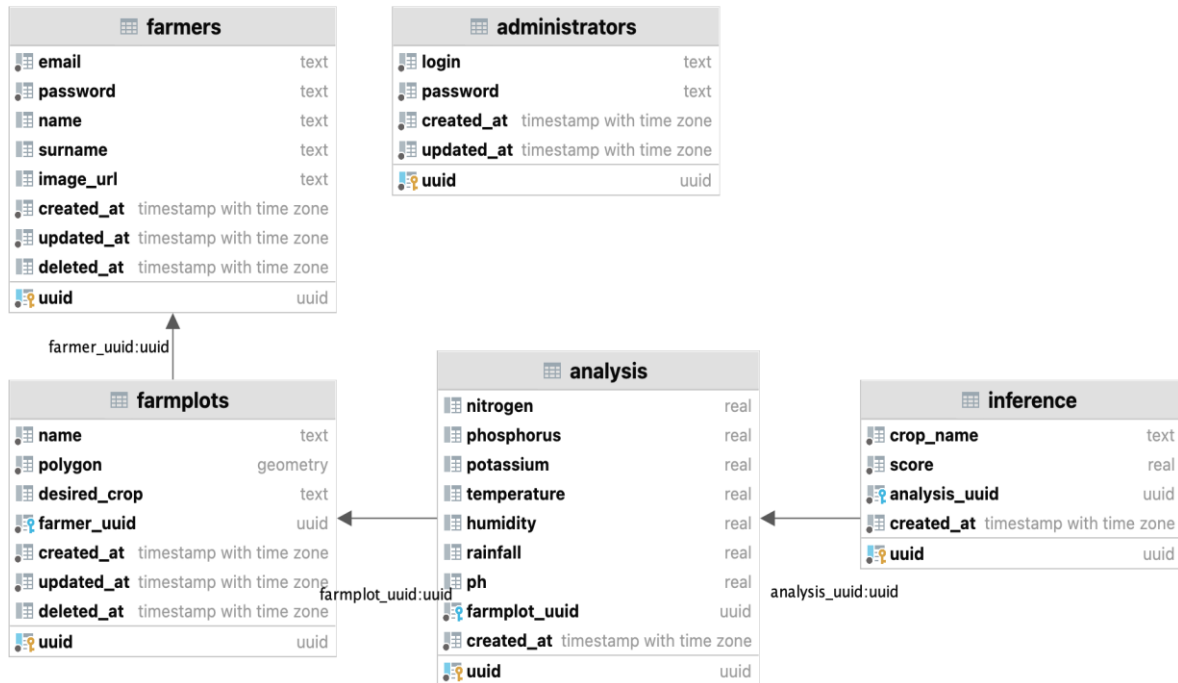


Рисунок 3.4 - Фізична модель бази даних

Після побудови фізичної моделі бази даних - наступний крок - це створення збережених процедур, тригерів для функціонування бази даних та зручної аналітики. Треба створити процедуру для збору статистики по проведеним аналізам та їх висновкам.

3.5 Тригери та збережені функції

В рамках розробки прототипа системи по планування та аналізу для отримання максимального врожаю з певної території в базі даних, окрім

стандартних функцій для запису і читання даних потрібно створити ще й функцію для вибірки необхідних даних для аналітики.

Для цього було створено три збережені функції:

- функцію, яка б по заданому ідентифікатору фермера повертала б нам інформацію про фермера, та його ділянки. Тут особливу увагу треба приділити тому щоб декодувати полігон з бінарних даних до координатного вигляду, щоб клієнтська частина могла коректно його відобразити. Також необхідно відобразити імена ділянок, та сільськогосподарські культури, які планується вирощувати;

- функцію, яка б вибирала по заданому ідентифікатору ділянки всі проведені аналізи по ній, зокрема дані по вмісту фосфору, азоту, вологості, температури, кількості опадів, кислотності ґрунту. До цих даних треба приєднати також результати з висновками по проведенім аналізам. Конкретно нас буде цікавити коефіцієнт придатності ділянки до вирощування тієї чи іншої сільськогосподарської культури;

- функцію, яка відповідала за внесення аналізів в базу даних. Справа в тому що для ділянки часто недостатньо зібрати одну пробу ґрунту і на неї покластися. Наприклад для ділянки більше 30 га, територію поділяють на 15-20 секторів (для степів під орні культури) та беруть проби з них [6]. Функція має приймати масиви значень і брати середнє для розрахунку.

Також в базі існують тригери для оновлення поля, відповідаючого за дату останнього оновлення поля. Функції для адміністратора для додавання нового фермера в систему.

3.6 Проєктування програмних засобів

Після побудови фізичної моделі бази даних слідує побудова програмного забезпечення. При розробці буде використовуватись методика об'єктно орієнтованого програмування. Nest.js фреймворк для платформи Node.js добре підтримує ООП, та має дуже потужні практики ін'єкції залежностей модулів. Такий підхід буде мати логічні переваги ООП в читанні коду, та підтримуючи практики ін'єкції залежностей ми розв'язуємо руки для кращого тестування та підстановки модулів в інші модулі, роблячи код легшим для модифікації та краще тестованим.

Класи будуть містити в собі роботу з базою даних, обробку серверний запитів, та роботу з третіми системами.

Діаграма класів – це структурна діаграма, що створюється в рамках мови UML, яка наочно показує ієрархію класів у системі, що розробляється, а також відображає їх атрибути, методи, інтерфейси та взаємозв'язки між ними.

На рисунку 3.5 зображена діаграма класів для модуля, що розробляється.

Оскільки архітектура програми клієнт-серверна, класи в клієнтському додатку призначені саме для відображення інформації, що зчитується з бази даних, включаючи процедури додавання, зміни та видалення інформації, а також вибірки певної інформації [12].

На діаграмі видно, що точкою входу до програми є клас App. Цей клас містить в собі інформацію про конфігурацію системи. Також він ініціює інші модулі і зв'язує їх у цілу систему. В системі є модулі авторизацій, модулі які презентують фермера, ділянку, аналізи і висновки.

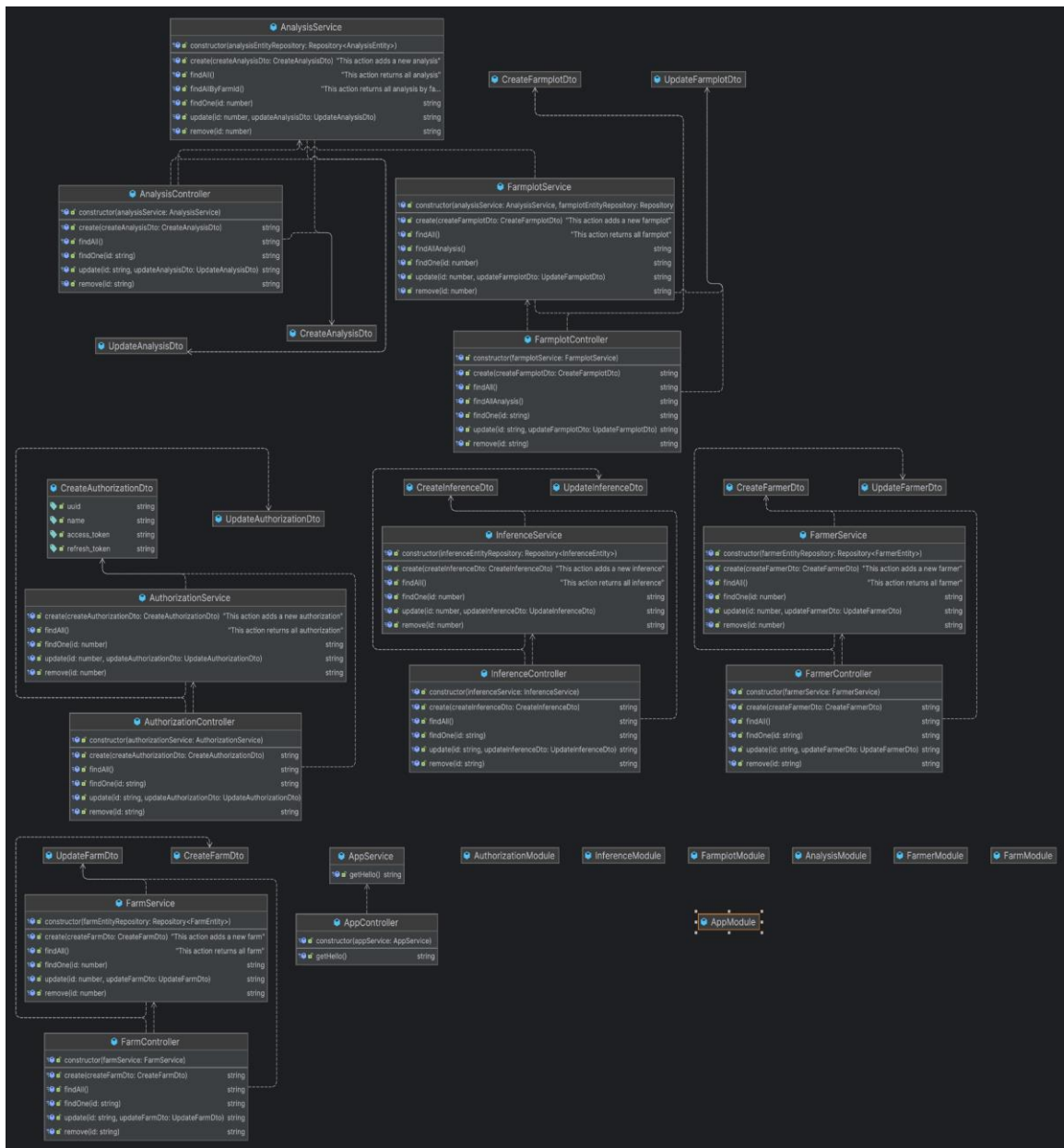


Рисунок 3.5 - діаграма класів і методів які приймають участь в системі

Ця діаграма відображає узагальнений функціонал розроблюваної системи. Проектування зроблено за допомогою фреймворка Nest.js для платформи Node.js, мовою TypeScript. Для генерації діаграм використано функціонал програми WebStorm.

Інтеграція з базою даних зроблена за допомогою ORM технології Typeorm, так як вона має багато модулів, спеціально спроектованих для найкращої роботи з

Nest.js. Ми можемо під'єднати окремий репозиторій до потрібного нам модуля та сервісу, та точно знати структуру таблиць з якими ми працюємо в даний момент.

```
import { Entity, Column, OneToOne, JoinColumn } from 'typeorm';
import { BaseEntity } from './base.entity';
import { FarmplotEntity } from './farmplot.entity';
import { InferenceEntity } from './inference.entity';

no usages  Dmitry Chernov

@Entity( options: { name: 'analysis' } )
export class AnalysisEntity extends BaseEntity {
  @Column( options: {type: 'real', nullable: true})
  phosphorus: number;

  @Column( options: {type: 'real', nullable: true})
  potassium: number;

  @Column( options: {type: 'real', nullable: true})
  ph: number;

  @Column( options: {type: 'real', nullable: true})
  nitrogen: number;

  @Column( options: {type: 'real', nullable: true})
  temperature: number;

  @Column( options: {type: 'real', nullable: true})
  humidity: number;

  @Column( options: {type: 'real', nullable: true})
  rainfall: number;

  @JoinColumn( options: { name: 'farmplot_uuid' })
  farmplot_uuid: string;

  @OneToOne( typeFunctionOrTarget: type => FarmplotEntity, inverseSide: farmplot : FarmplotEntity => farmplot.analysis )
  farmplot: FarmplotEntity;

  @OneToOne( typeFunctionOrTarget: type => InferenceEntity, inverseSide: inference : InferenceEntity => inference.analysis )
  inference: InferenceEntity;
}
```

Рисунок 3.6 - приклад створення сутності за допомогою інструментів Typeorm.

Для створення ділянки фермером була розроблена клієнтська частина на Next.js для бібліотеки React. Вона включає в себе сторінки авторизації, створення ферми та визначення її меж, введення результатів аналізів.

Процес створення ділянки показано рисунку 3.7

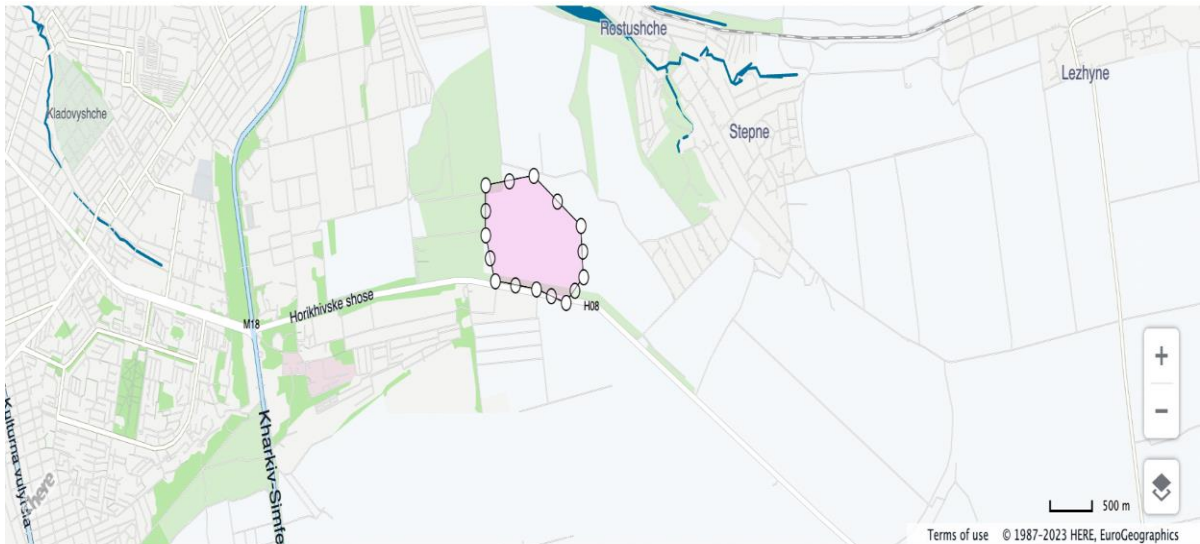


Рисунок 3.7 - створення ділянки на мапі від HERE Maps

Після створення ділянки на клієнтській частині ми надсилаємо запит на збереження до серверу на контроллер який відповідає за ділянку. Сервер використовуючи API для openweathermap бере дані про опади та температуру для заданої ділянки. А після цього, використовуючи SoilGrids API виконує запит за характеристиками ґрунту по координатам. Якщо дані не надходять - виводимо користувачу (фермеру) форму для заповнення даних по аналізу.

Коли в системі вже є повна інформація по ділянці - ми виконуємо запит до тренованої нейромережі і отримуємо інформацію про найбільш підходящу культуру для ділянки. Нижче наведений приклад висновку.

Для першого виклику використовувались аргументи

nitrogen=65

phosphorus=60

potassium=22

temperature=25.88

humidity=72.52

ph=6.60

rainfall=20.93

```

14 # Завантаження даних
15 df = pd.read_csv('Crop_Recommendation.csv')
16
17 # Перетворення міток
18 labelencoder = LabelEncoder()
19 df['label_codes'] = labelencoder.fit_transform(df['label'])
20
21 # Розділення даних
22 X = df[['N', 'P', 'K', 'Temperature', 'Humidity', 'ph', 'Rainfall']]
23 y = pd.get_dummies(df['label_codes'])
24
25 # Завантаження збереженої моделі
26 with open('RandomForest.pkl', 'rb') as file:
27     multi_target_forest = pickle.load(file)
28 # Тестування для конкретного випадку
29 test_input = {
30     'N': [nitrogen],
31     'P': [phosphorus],
32     'K': [potassium],
33     'Temperature': [temperature],
34     'Humidity': [humidity],
35     'ph': [ph],
36     'Rainfall': [rainfall]
37 }
38 test_df = pd.DataFrame(test_input)
39 test_predicted = multi_target_forest.predict(test_df)
40
41 # Отримання назви рекомендованої культури
42 predicted_cultures = labelencoder.inverse_transform(np.where(test_predicted[0] == 1)[0])
43 recommended_culture = predicted_cultures[0] if len(predicted_cultures) > 0 else "Не знайдено підходящих культур"
44 print("Рекомендована культура:". recommended_culture)

```

Run test x

/Users/dchernov/PycharmProjects/pythonProject/.venv/bin/python /Users/dchernov/PycharmProjects/pythonProject/test.py

Рекомендована культура: maize

Process finished with exit code 0

Рисунок 3.8 - результат відпрацювання нейромережі з першим набором аргументів

3.7 Тестування на різних даних

Для тестування нейромережі будуть вибрані дані, які не співпадають на 100% з тренувальними даними. Нижче наведена таблиця з тестами для визначення результату.

Таблиця 3.8 - отримані значення для тестування аналізу і результати

Nitrogen	Phosphorus	Potassium	Temperature	Humidity	Ph	Rainfall	Результат
6	139	199	25.67	81.621	6.4	74.109	Виноград
90	16	45	24.9	80	6.2	50.5	Кавун
10	125	196	22.31	90.03	5.7	113.1	Яблуко
21	64	17	15.7	19	5.9	108	Квасоля

Нижче наведені рисунки 3.9, 3.10, 3.11, та 3.12 з відповідними тестами нейромережі в системі.

```
test_predicted = multi_target_forest.predict(test_df)

# Отримання назви рекомендованої культури
predicted_cultures = labelencoder.inverse_transform(np.where(test_predicted[0] == 1)[0])
recommended_culture = predicted_cultures[0] if len(predicted_cultures) > 0 else "Не знайдено підходящих культур"
print("Дані для тесту", test_input)
print("Рекомендована культура:", recommended_culture)
```

test x

```
/Users/dchernov/PycharmProjects/pythonProject/.venv/bin/python /Users/dchernov/PycharmProjects/pythonProject/test.py
Дані для тесту {'N': [6], 'P': [139], 'K': [199], 'Temperature': [25.67], 'Humidity': [81.621], 'ph': [6.4], 'Rainfall': [74.109]}
Рекомендована культура: grapes
```

Рисунок 3.9 - перший тест. Виноград

```
# Отримання назви рекомендованої культури
predicted_cultures = labelencoder.inverse_transform(np.where(test_predicted[0] == 1)[0])
recommended_culture = predicted_cultures[0] if len(predicted_cultures) > 0 else "Не знайдено підходящих культур"
print("Дані для тесту", test_input)
print("Рекомендована культура:", recommended_culture)
```

test x

```
/Users/dchernov/PycharmProjects/pythonProject/.venv/bin/python /Users/dchernov/PycharmProjects/pythonProject/test.py
Дані для тесту {'N': [90], 'P': [16], 'K': [45], 'Temperature': [24.9], 'Humidity': [80], 'ph': [6.2], 'Rainfall': [50.5]}
Рекомендована культура: watermelon
```

Рисунок 3.10 - другий тест. Кавун

```
# Отримання назви рекомендованої культури
predicted_cultures = labelencoder.inverse_transform(np.where(test_predicted[0] == 1)[0])
recommended_culture = predicted_cultures[0] if len(predicted_cultures) > 0 else "Не знайдено підходящих культур"
print("Дані для тесту", test_input)
print("Рекомендована культура:", recommended_culture)
```

test x

:

/Users/dchernov/PycharmProjects/pythonProject/.venv/bin/python /Users/dchernov/PycharmProjects/pythonProject/test.py
Дані для тесту {'N': [10], 'P': [125], 'K': [196], 'Temperature': [22.31], 'Humidity': [90.03], 'ph': [5.7], 'Rainfall': [113.1]}
Рекомендована культура: apple

Рисунок 3.11 - третій тест. Яблуко

```
# Отримання назви рекомендованої культури
predicted_cultures = labelencoder.inverse_transform(np.where(test_predicted[0] == 1)[0])
recommended_culture = predicted_cultures[0] if len(predicted_cultures) > 0 else "Не знайдено підходящих культур"
print("Дані для тесту", test_input)
print("Рекомендована культура:", recommended_culture)
```

test x

:

/Users/dchernov/PycharmProjects/pythonProject/.venv/bin/python /Users/dchernov/PycharmProjects/pythonProject/test.py
Дані для тесту {'N': [10], 'P': [125], 'K': [196], 'Temperature': [22.31], 'Humidity': [90.03], 'ph': [5.7], 'Rainfall': [113.1]}
Рекомендована культура: apple

Рисунок 3.12 - четвертий тест. Квасоля

Ці тести спрямовані на визначення точності визначення найбільш підходящої культури для заданої ділянки. І ці тести показали, що нейромережа видає результати згідно з даними наданими для тренування. Тобто тест можна вважати успішним.

ВИСНОВКИ

Дослідивши існуючі методи визначення якості та параметрів землі - визначили, що є критично важливим для вибору відповідних культур у сільському господарстві. Також аналіз показує, що хоча існуючі методики є різноманітними та широко застосовуваними, вони далекі від ідеалу. Основна проблема полягає у високій вартості та складності проведення аналізів, що робить їх недоступними для багатьох дрібних та починаючих фермерів.

Ця робота підкреслює необхідність розробки програмного забезпечення, яке б дозволяло фермерам з легкістю оцінювати характеристики своєї землі без потреби витрачати значні кошти на аналізи. Така система мала б включати інтелектуальні алгоритми та нейромережі для аналізу даних, які б забезпечували точне визначення вмісту поживних речовин, вологості ґрунту та інших ключових параметрів, що впливають на врожайність та якість культур.

Особлива увага приділена визначенню оптимальних умов для конкретних культур бо недостатньо зрозуміти наскільки гарний цей ґрунт а абсолюті. Більше треба фокусуватись на конкретних потребах конкретних рослин. Це включає оцінку якості ґрунту, забезпечення необхідного рівня поживних речовин та води, що має важливе значення для досягнення високих показників урожайності.

Завершуючи, можна сказати, що розробка програми є важливим кроком до підвищення продуктивності та стійкості сільського господарства. Це дозволить фермерам ефективно управляти своїми ресурсами, вибирати найкращі культури для своїх земель і в цілому сприяти сталому розвитку агропромислового сектору.

Маючи за мету створити систему надійною та продуктивною одночасно і не витратити при цьому надзвичайно багато часу було проведено дослідження технологій на ринку.

Обираючи платформу яка б найбільше підходила під поставлену мету, технології для серверної частини, бази даних, та клієнтської частини - було обрано оптимальні, як на мій погляд технології.

Ці технології дають можливість сконцентруватися на логіці системи і не продумувати дуже багато реалізацію базових речей, таких як структуру даних, алгоритми і базові контракти для комунікацій між вузлами системи.

Використовуючи необхідні технології, було досягнуто мети і побудовано модель системи, яка б розраховувала найбільш оптимальну культуру для заданої сільськогосподарської ділянки, використовуючи дані з супутників, погодних даних з інтернету та введені вручну аналітичні дані.

Проведено тестування створеної системи і перевірено вихідні дані. Проведене тестування показало правильність роботи системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. “Вплив кислотності та засоленості ґрунту на врожай пшениці.” Журнал Агроном, 13 July 2022, <https://www.agronom.com.ua>. Accessed 22 October 2023.
2. “Значення органічних речовин ґрунту та добрив для живлення рослин.” AgroBio, <https://agro.bio>. Accessed 25 October 2023.
3. Agrohimanaliz, 2 October 2022, <https://agrohimanaliz.com.ua/> Accessed 4 November 2023.
4. “Водний режим ґрунту на посівах пшениці озимої в умовах Південного Степу.” Агробізнес сьогодні, 7 October 2016, <https://agro-business.com.ua>. Accessed 4 November 2023.
5. “Проектування бази даних — Вікіпедія.” Вікіпедія, https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85. Accessed 7 December 2023.
6. Григора, Татяна. Відбір проб ґрунту для хімічного аналізу, 12 April 2019, <https://himanaliz.ua/uk/vidbir-prob-gruntu-dlya-khimichnogo-analiza/>. Accessed 8 December 2023.
7. Раттан, Лал. “Regenerative agriculture for food and climate.” *Journal of Soil and Water Conservation*, vol. 75, no. 1, 2020, pp. 123A–124A.
8. Кук, Джеймс, and Вашингтонський державний університет. Сільськогосподарська дослідницька служба. -, 2016.
9. “Методи і засоби агрометеорологічних вимірювань параметрів ґрунтів, прилад ВПГ-4ц.” Український гідрометеорологічний інститут, <https://uhmi.org.ua>. Accessed 27 October 2023.

10. “About.” PostgreSQL, <https://www.postgresql.org/about/>. Accessed 25 November 2023.
11. “How to Choose the Right Backend Technology for Your App.” Async Labs, 21 May 2021, <https://www.asynclabs.co/blog/software-development/how-to-choose-the-right-backend-technology-for-your-app/>. Accessed 21 November 2023.
12. “Introduction.” HERE Technologies, 27 November 2023, <https://www.here.com/docs/bundle/vector-tile-api-developer-guide/page/README.html>. Accessed 7 December 2023.
13. Priya, Deva. “10 Essential Backend Technologies for Web Development.” Analytics Insight, 26 September 2023, <https://www.analyticsinsight.net/10-essential-backend-technologies-for-web-development/>. Accessed 21 November 2023.
14. Raval, Nihar, and Milit Panchasara. “React vs Angular: Which One is Best for Your Next Front-end Project?” Radixweb, 8 November 2023, <https://radixweb.com/blog/react-vs-angular>. Accessed 25 November 2023.
15. Smallcombe, Mark. “PostgreSQL vs MySQL: The Critical Differences.” Integrate.io, 20 September 2023, <https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/>. Accessed 25 November 2023.
16. “SoilGrids API User Guide — Cool Farm Tool Online 2.8.0 documentation.” Cool Farm Tool, https://app.coolfarmtool.org/docs/api/v1/soil_grids/soil_grids.html. Accessed 9 December 2023.
17. Zunker, F. Das verhalten des Bodens zum Wasser. Handbuch der Bodenlehre. 6 ed., vol. 1, Berlin, 1930.

Додаток А

ЗАПИТИ ДО СКБД ДЛЯ СТВОРЕННЯ БАЗИ ДАНИХ

Лістинг А.1 - Файл “dump.sql”

```

CREATE TABLE IF NOT EXISTS administrators (
    uuid uuid primary key unique not null DEFAULT
gen_random_uuid(),
    login text not null,
    password text not null,
    created_at timestamptz not null default CURRENT_TIMESTAMP,
    updated_at timestamptz not null default CURRENT_TIMESTAMP
);

CREATE FUNCTION update_updated_on()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = now();
    RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER adminisrator_on_update
BEFORE UPDATE ON administrators
FOR EACH ROW
EXECUTE PROCEDURE update_updated_on();

-- farmer
CREATE TABLE IF NOT EXISTS farmers (
    uuid uuid primary key unique not null DEFAULT
gen_random_uuid(),
    email text not null,
    password text not null,
    name text,
    surname text,
    image_url text,
    created_at timestamptz not null default CURRENT_TIMESTAMP,
    updated_at timestamptz not null default CURRENT_TIMESTAMP,
    deleted_at timestamptz
);

CREATE TRIGGER farmers_on_update
BEFORE UPDATE ON farmers
FOR EACH ROW
EXECUTE PROCEDURE update_updated_on();

```

```

-- farmplot
CREATE EXTENSION postgis;

CREATE TABLE IF NOT EXISTS farmplots (
    uuid uuid primary key unique not null DEFAULT
gen_random_uuid(),
    name text not null ,
    polygon geometry not null ,
    desired_crop text,
    farmer_uuid uuid references farmers(uuid) not null,
    created_at timestamptz not null default CURRENT_TIMESTAMP,
    updated_at timestamptz not null default CURRENT_TIMESTAMP,
    deleted_at timestamptz
);

CREATE TRIGGER farmplots_on_update
BEFORE UPDATE ON farmplots
FOR EACH ROW
EXECUTE PROCEDURE update_updated_on();

-- analysis
CREATE TABLE IF NOT EXISTS analysis (
    uuid uuid primary key unique not null DEFAULT
gen_random_uuid(),
    nitrogen real,
    phosphorus real,
    potassium real,
    temperature real,
    humidity real,
    rainfall real,
    ph real,
    farmplot_uuid uuid references farmplots(uuid) not null,
    created_at timestamptz not null default CURRENT_TIMESTAMP
);

-- inference
CREATE TABLE IF NOT EXISTS inference (
    uuid uuid primary key unique not null DEFAULT
gen_random_uuid(),
    crop_name text not null,
    score real not null,
    analysis_uuid uuid references analysis(uuid) not null,
    created_at timestamptz not null default CURRENT_TIMESTAMP
);

```