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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.42_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система з підтримки обприскування сільськогосподарських полів за допомогою БПЛА»

Виконав:
студент 2 курсу, групи ІК-21мп
Коваленко Микола Юрійович _____

Керівник:
доц. кафедри ІСТ, к.т.н., доц.,
Крилов Євген Володимирович _____

Рецензент:
доц. кафедри ІПІ, к.т.н., доц.,
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Коваленко Микола Юрійович

1. Тема дисертації «Інформаційна система з підтримки обприскування сільськогосподарських полів за допомогою БПЛА», науковий керівник дисертації Крилов Євген Володимирович, к.т.н., доц. кафедри ІСТ ФІОТ, затверджені наказом по університету від «07» 11 2023 р. № 5168-с.

2. Термін подання студентом дисертації «08» 01 2024 р.

3. Об’єкт дослідження: інформаційна система підтримки обприскування за допомогою БПЛА.

4. Вихідні дані: статистичні відомості про кількість сільськогосподарських підприємств Україні, їхні обсяги виробництва, витрат та прибутків.

5. Перелік завдань, які потрібно розробити: проаналізувати область об’єкту дослідження; проаналізувати існуючі рішення; сформувати задачу роботи; спроєктувати архітектуру системи; реалізувати інформаційну систему; розробити стартап-проєкт.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма прецедентів, розширена діаграма прецедентів, діаграма активності, структурна схема системи, ER діаграма, фізична модель даних, блок-

схеми побудови траєкторії обходу покриття та процесу розбиття робочої області на сітку.

7. Орієнтовний перелік публікацій: Коваленко М. Ю. Аналіз можливості використання інформаційних систем обприскувальних БПЛА як методу відновлення сільськогосподарських потужностей України / М. Ю. Коваленко, Є. В. Крилов // ІІІ Міжнародна спеціалізована наукова конференція «Інноваційні тенденції сьогодення в сфері природничих, гуманітарних та точних наук», Рівне, 2023. С. 98-99.

9. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд літератури. Виявлення основної проблематики реалізації	08.09	
2	Вивчення існуючих рішень. Аналіз їх переваг та недоліків	15.09	
3	Постановка задачі. Вибір технічних рішень реалізації	29.09	
4	Формування вимог та критеріїв системи, що розробляється	06.10	
5	Розробка системи	24.11	
6	Тестування системи із застосуванням симуляції реальних умов	01.12	
7	Оформлення документації	18.12	
8	Подання роботи на попередній захист	19.12	
9	Подання роботи на основний захист	08.01	

Студент Микола КОВАЛЕНКО

Науковий керівник Євген КРИЛОВ

РЕФЕРАТ

Пояснювальна записка до магістерської дисертації на тему «Інформаційна система з підтримки обприскування сільськогосподарських полів за допомогою БПЛА» містить 143 с., 24 табл., 52 рис., 9 дод., 29 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, СІЛЬСЬКЕ ГОСПОДАРСТВО, БПЛА, ДРОН, DYNAMODB, ТРАЄКТОРІЯ ОБХОДУ ПОКРИТТЯ.

Об’єктом дослідження є процес обприскування сільськогосподарських полів. Предметом дослідження же виступає інформаційна система підтримки обприскування за допомогою БПЛА.

Метою роботи є розробка інформаційної системи, яка сприятиме вдосконаленню та оптимізації процесів обприскування сільськогосподарських полів за допомогою БПЛА.

Робота є актуальною, оскільки вона спрямована на розробку інноваційного інструменту, який допоможе оптимізувати та поліпшити процеси ведення сільського господарства, частка якого у загальному виробництві України є найбільшою.

Результатом роботи є інформаційна система з підтримки обприскування сільськогосподарських полів за допомогою БПЛА, яка використовує спеціально розроблені алгоритми для побудови оптимальної траєкторії та базу даних DynamoDB для досягнення високої швидкодії.

ABSTRACT

The explanatory note to the master's thesis on the topic "Information support system for spraying of agricultural fields using UAVs" contains 143 p., 24 tab., 52 draw., 9 app., 29 sources.

INFORMATION SYSTEM, AGRICULTURE, UAV, DRONE, DYNAMODB, COVERAGE TRAJECTORY.

The object of research is the process of spraying agricultural fields. The subject of the research is the information system for spraying support using UAVs.

The purpose of the work is the development of an information system that will contribute to the improvement and optimization of the processes of spraying agricultural fields with the help of UAVs.

The work is relevant, as it is aimed at the development of an innovative tool that will help optimize and improve the processes of agriculture, the share of which in the total production of Ukraine is the largest.

The result of the work is an information system to support the spraying of agricultural fields with the help of UAVs, which uses specially developed algorithms to build the optimal trajectory and the DynamoDB database to achieve high performance.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 9
ВСТУП 10
1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ОБПРИСКУВАННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПОЛІВ 12
1.1 Традиційні методи обприскування, їх переваги та недоліки 12
1.1.1 Ручне обприскування 13
1.1.2 Тракторне обприскування 14
1.1.3 Обприскування літальними апаратами 15
1.1.4 Обприскування через системи зрошення 17
1.2 Обприскування за допомогою БПЛА, його переваги та недоліки 19
1.3 Визначення проблеми та постановка задачі 25
Висновки до розділу 1 29
2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 30
2.1 Основні види продукції з обприскування за допомогою БПЛА на ринку 30
2.1.1 Drone Survey Services 30
2.1.2 Auto Spray Systems 32
2.1.3 Hylio AgroSol 33
2.1.4 Argemo 35
2.2 Формування вимог до системи 39
Висновки до розділу 2 40
3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ 41
3.1 Загальна архітектура системи 41
3.1.1 Монолітна архітектура 41
3.1.2 Мікросервісна архітектура 43
3.1.3 Клієнт-серверна архітектура 45
3.1.4 Архітектура, керована подіями (EDA) 47
3.1.5 Вибір архітектури для інформаційної системи з підтримки обприскування за допомогою БПЛА 50

3.2 Опис функціональної складової системи 54
3.3 Послідовність взаємодії користувачів з системою 59
3.4 Засоби розробки системи 61
3.5 Вибір бази даних 70
3.6 Методи побудови траєкторії обходу покриття 73
Висновки до розділу 3 82
4 РЕАЛІЗАЦІЯ СИСТЕМИ 84
4.1 Опис структури системи 84
4.1.1 API контролери 85
4.1.2 Сервіси 87
4.1.3 Моделі 92
4.1.4 База даних 92
4.2 Реалізація алгоритму побудови траєкторії обходу покриття 95
4.3 Реалізація інтерфейсів користувача 99
Висновки до розділу 4 114
5 РОЗРОБКА СТАРТАП-ПРОЄКТУ 115
5.1 Опис ідеї проєкту 115
5.2 Технологічний аудит ідеї проєкту 118
5.3 Аналіз ринкових можливостей запуску стартап-проєкту 119
5.4 Розроблення ринкової стратегії проєкту 128
5.5 Розроблення маркетингової програми стартап-проєкту 132
Висновки до розділу 5 136
ВИСНОВКИ 138
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 140
ДОДАТОК А. Діаграма прецедентів ..... **Error! Bookmark not defined.**
ДОДАТОК Б. Розширена діаграма прецедентів ..... **Error! Bookmark not defined.**
ДОДАТОК В. Діаграма активності ..... **Error! Bookmark not defined.**
ДОДАТОК Г. Структурна схема системи ..... **Error! Bookmark not defined.**
ДОДАТОК Д. ER діаграма ..... **Error! Bookmark not defined.**
ДОДАТОК Е. Діаграма класів ..... **Error! Bookmark not defined.**

ДОДАТОК Ж. Узагальнена блок-схема процесу побудови траєкторії обходу покриття ..... **Error! Bookmark not defined.**
ДОДАТОК И. Блок-схема процесу розбиття РО на сітку**Error! Bookmark not defined.**
ДОДАТОК К. Посилання на GitHub з кодом системи 144

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БПЛА – безпілотний літальний апарат;

ШІ – штучний інтелект;

РО – робоча область;

ТЗ – точка завантаження;

БД – база даних;

РП – робочий план;

ТОП – траєкторія обходу покриття;

API – application programming interface;

EDA – event-driven architecture;

P2P –peer-to-peer;

DoS – denial of service;

MITM – man in the middle;

ПК – персональний комп’ютер;

SQL – structured query language;

UML – unified modeling language;

MVC – model-view-controller;

GSI – global secondary index;

ER – entity relationship;

UI – user interfaces;

UX – user experience;

HTTP – hypertext transfer protocol;

HTTPS – hypertext transfer protocol secure;

SHA – secure hash algorithm.

ВСТУП

Сучасний світ інформаційних технологій постійно розвивається і не стоїть на місці. Кожного року з’являються все нові й нові рішення, які привносять покращення в майже усі галузі діяльності людини. Не є виключенням й участь сільського господарства у впровадженні інформаційних технологій.

Сільське господарство є однією з найважливіших галузей глобальної економіки, забезпечуючи нашу планету необхідними продуктами харчування. Щороку виробництво сільськогосподарських культур стає все більшим завданням, оскільки населення Землі зростає, і так само зростають вимоги до безпеки та якості продуктів. В умовах сучасного аграрного господарства великий акцент робиться на оптимізації процесів та ресурсів для досягнення високої продуктивності та стійкості до зовнішніх викликів.

За останні десятиліття обробка сільськогосподарських полів за допомогою безпілотних літальних апаратів (БПЛА) стала дедалі більш популярною та ефективною практикою в аграрному секторі. Використання БПЛА дозволяє здійснювати обприскування полів з високою точністю, зменшуючи витрати на пальне, хімічні реагенти та зменшуючи негативний вплив на навколишнє середовище. Однак ефективне впровадження та використання цієї технології вимагає розробки та впровадження спеціалізованих інформаційних систем, призначених для підтримки обприскування сільськогосподарських полів за допомогою БПЛА.

Дана робота є актуальною та важливою, оскільки вона спрямована на розробку інноваційних інструментів, які допоможуть оптимізувати та поліпшити процеси сільського господарства. Тема роботи передбачає аналіз і розробку інформаційних систем, які не лише підтримують обприскування полів за допомогою БПЛА, але й враховують низку факторів, таких як погодні умови, різноманітність культур, вимоги до ефективності та екологічної безпеки. Дослідження в цій області має великий потенціал для розвитку

сільськогосподарського виробництва та зменшення його негативного впливу на навколишнє середовище.

Також не слід забувати і про потенційний вплив подібної системи на повоєнну відбудову України. Оскільки використання БПЛА для обприскування є менш витратним у плані грошових та фізичних ресурсів, а також точнішим та безпечнішим як для навколишнього середовища, так і для працівників, тоді цей спосіб може стати проривним у відновленні сільськогосподарських земель, що були тим чи іншим чином пошкодженні під час війни.

В такому разі об’єктом дослідження цієї роботи є процес обприскування сільськогосподарських полів. Предметом дослідження же виступає інформаційна система підтримки обприскування за допомогою БПЛА.

Метою ж цієї роботи є розробка інформаційної системи, яка сприятиме вдосконаленню та оптимізації процесів обприскування сільськогосподарських полів за допомогою БПЛА.

Пояснювальна записка до магістерської дисертації складається з п’яти розділів. У першому розділі проводиться аналіз сучасних методів обприскування сільськогосподарських полів, огляд їхніх переваг та недоліків, визначення проблеми та постановка задачі. У другому розділі відбувається аналіз існуючих на ринку рішень, визначаються їхні переваги та недоліки, на основі яких формуються функціональні вимоги до майбутньої системи. Третій розділ присвячений проєктуванню системи, і в ньому розглядаються види архітектурних патернів, проводиться вибір архітектури, яка відповідатиме визначеним раніше вимогам, описується функціональна складова системи, визначається порядок взаємодії користувачів із системою, визначаються основні технології, які будуть використані під час розробки та роботи системи, а також відбувається проєктування бази даних. У четвертому розділі розкривається процес реалізації системи з акцентом на розробці алгоритму побудови траєкторії обходу покриття та користувацьких інтерфейсів. П’ятий розділ присвячений розробці стартап-проєкту за тематикою магістерської дисертації.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ОБПРИСКУВАННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПОЛІВ

1.1 Традиційні методи обприскування, їх переваги та недоліки

Обприскування полів є хоч і не найголовнішим, але все ж доволі важливим аспектом сучасного сільського господарства. Сам процес обприскування є доволі простим за визначенням – нанесення дрібних крапель рідких хімічних речовин на рослини та ґрунт. Метою ж цього процесу може бути як внесення добрив, так і захист від шкідників, бур’янів та різноманітних захворювань.

Обприскування може проводитися за допомогою різних способів в залежності від багатьох факторів, таких як: об’єм роботи, тип культури, яка потребує обприскування, доступність технічних засобів. Загалом же, методів обприскування існує не так вже й багато, а ефективних – ще менше. Найпопулярнішими наразі можна вважати наступні:

– ручне обприскування – використовує розраховані на носіння людиною переносні обприскувачі невеликого об’єму для розпилення хімічних сполук. Зазвичай використовується в малих господарствах або для роботи в місцях з обмеженим доступом для механізованої техніки;

– тракторне обприскування – використовує трактори або іншу механізовану техніку. Метод є доволі ефективним при обробці великих площ, і наразі являється найпопулярнішим в усьому світі у зв’язку з оптимальним співвідношенням ціни, якості та доступності;

– обприскування літальними апаратами – передбачає використання спеціально обладнаних літальних апаратів – літаків чи гелікоптерів – які можуть використовуватися для обприскування великих аграрних земельних ділянок. Метод є доволі ефективним в плані точності, але досить важкодоступним, оскільки потребує спеціальної інфраструктури;

– обприскування через системи зрошення – метод, який використовує системи поливу для розповсюдження хімічних речовин. Можуть бути використані системи спрейового, дощувального та крапельного зрошення. Такий метод може

бути використаний лише на невеликих ділянках та лише з урахуванням особливостей кожного з видів систем.

Розглянемо кожен з методів більш детально.

1.1.1 Ручне обприскування

Ручне обприскування є найбільш примітивним методом розпилення хімічних речовин. Метод передбачає використання людьми спеціальних переносних приладів-обприскувачів, які використовують ручний насос, за допомогою котрого в бак з хімічною речовиною закачується повітря, створюючи збільшений тиск. Після чого людина-оператор, натискаючи кнопку (або відкриваючи кран) подачі рідини, вивільняє вміст бака, який за рахунок підвищеного тиску поступає зі збільшеною швидкістю через шланг. Проходячи через спеціальну насадку, призначенням якої є дисперсія потоку рідини, хімічна речовина розпиляється на невелику площу, на яку вказує оператор.

До переваг цього методу можна віднести:

– гнучкість використання – за допомогою ручного обприскування дуже зручно обробляти окремі невеликі ділянки;

– можливість застосування у важкодоступних для техніки місцях – не завжди великогабаритна техніка може дістатися абсолютно усіх куточків ділянки, що підлягає обробки. Особливим випадком є горбистий ландшафт, який може заважати проходу техніки, а от людина з ручним обприскувачем може ним пересуватися;

– доступність – ручне обприскування є найбільш доступним серед усіх видів обприскування. Це пояснюється доволі низькою вартістю приладів ручного обприскування, і як результат, їхньою масовою розповсюдженістю.

До недоліків же ручного обприскування можна віднести:

– низька продуктивність – з усіх перелічених методів обприскування ручне є найменш ефективним;

– висока трудомісткість – метод може бути дуже затратним за часом, особливо за обробки великих площ, та використанням людських ресурсів.

1.1.2 Тракторне обприскування

На відміну від загальної назви даний метод включає обприскування не тільки за допомогою трактора. Основним і найпоширенішим способом обприскування дійсно є використання трактора із застосуванням додаткового обладнання – навісного чи причіпного обприскувача. Але також має місце і застосування спеціальних самохідних колісних обприскувальних установок. Та головний принцип методу є незмінним – обприскування проводиться за допомогою великогабаритної колісної сільськогосподарської техніки.

Обприскування цим методом має свій перелік переваг:

– швидкість – за рахунок великого об’єму резервуара з рідиною та широкої площі покриття вдається досягти відносно швидкого оброблення, бо необхідність дозаправки виникає набагато рідше ніж за використання інших методах;

– доступність – хоч необхідне обладнання і не є настільки ж поширеним та дешевим як у випадку з ручним обприскування, втім воно все ще доволі доступне і майже завжди його можна знайти на ринку в будь-якій країні;

– сумісність – найчастіше використовуються саме причіпні обприскувачі, які можуть кріпитися до іншої сільськогосподарської технікі (зазвичай до тракторів), що вже використовується в господарстві. Тому не виникає необхідності купувати цілі окремий засіб для обприскування, а також в разі виходу з ладу одного трактора обприскувач можна причепити до іншого;

– низька залежність від погодних умов – даний метод не є надто залежним від сильного вітру чи дощу. Звісно, обприскування під час дощу має небагато сенсу, а от сильний вітер може перешкодити таким методам як обприскування літальними апаратами (літаками чи БПЛА).

Якщо підсумувати усі переваги тракторного обприскування, то стає зрозуміло, чому цей метод є найпопулярнішим. Навіть за наявності певних

недоліків, він однаково залишається оптимальним за співвідношенням доступності та продуктивності.

Втім незважаючи на значні переваги, під час вибору методу обприскування слід враховувати і його недоліки, і орієнтуватися на особливості власного господарства, а не на загальний тренд. До списку недоліків тракторного методу обприскування можна віднести:

– погана прохідність – великі габарити техніки хоч і є перевагою, коли йдеться про об’єм резервуара та широку смугу покриття, але вони також є і завадою, якщо необхідно обробити важкодоступні ділянки, або коли ділянка знаходиться на схилах горбистої місцевості;

– низька ефективність – метод не є оптимальним з точки зору об’єму рідини, який використовується. Враховуючи це, а також низьку якість розпилення, ефективність хімічних речовин є доволі низькою. Так лише близько 30% хімічної речовини використовується ефективно, а інші ж 70% не завдають значної шкоди шкідникам та не формують захисного шару на рослинах, але завдають серйозного забруднення навколишньому середовищу [1];

– ймовірність заподіяння шкоди рослинам – оскільки обприскування найчастіше відбувається на невеликій відстані від самих рослин, то підвищується ризик їхнього фізичного травмування, а також ризик хімічного отруєння;

– пошкодження рослин на шляху коліс – використання колісної техніки призводить до формування смуг з пошкоджених рослин на шляху, де проходять колеса. Така шкода є неминучим наслідком використання цього методу, і не завжди рослинам, які були уражені таким чином під час обприскування, вдається відновитися.

1.1.3 Обприскування літальними апаратами

Даний метод обприскування визначний тим, що він передбачає використання найчастіше літаків, які мають вбудовані резервуари для хімічних речовин та системи розпилення цих речовин. За рахунок висоти польоту вдається

досягти широкого покриття за один проліт над ділянкою, тому для повного покриття необхідно не так багато часу. В залежності від розміру ділянки та розміру самого літального апарату здійснити повне обприскування усього поля можна за одну чи дві заправки. Втім використання літаків може бути обмежене багатьма факторами, які слід враховувати під час розгляду цього методу.

До переваг можна віднести такі:

– швидкість – за рахунок великої площі покриття та високої швидкості самих літаків вдається досягти найшвидших показників обприскування серед усіх методів;

– великий об’єм резервуара – дозволяє робити небагато дозаправок, або ж не робити їх взагалі;

– точність та рівномірність розпилення – за рахунок висоти випускання речовини вдається досягти рівномірного розпилення на всій робочій ділянці, а сучасні системи дозування роблять процес розпилення доволі точним;

– застосування у важкодоступних місцях – використання літального апарату нівелює вплив рельєфу на час та якість обприскування, а також літак не є залежним від прохідності на тій чи іншій ділянці, а тому може доставляти вантаж майже до будь-якої необхідної точки.

Але попри усі зазначені переваги використання літальних апаратів має ряд значних обмежень, які не дозволяють цьому методу стати найбільш популярним. До подібних недоліків належать:

– висока вартість – далеко не кожне господарство може дозволити собі мати літак для обприскування, або хоча б орендувати на певний період, бо ціни на подібні послуги є набагато вищими за інші методи, а ціна техніки та її обслуговування є неспівставно великою;

– залежність від наявної авіаційної інфраструктури – якщо господарство і може собі дозволити придбання необхідного літального апарату, тоді для нього необхідно також мати споряджений аеродром, що додає забагато нових клопотів. Якщо ж планувати орендувати послуги з обприскування літаком, то у найближчій від господарства місцевості має бути присутній аеродром та компанія (або інше

господарство), яка готова буде надати такі послуги, що не є доволі розповсюдженим випадком, а радше навіть навпаки – виключенням з правил;

– залежність від погодних умов – висота розпилення є як перевагою цього методу, так і його недоліком. Навіть невеликий вітерець здатний значно ускладнити процес розпилення. Сильний же вітер може не тільки здути саму розпилену речовину, а й ускладнити керування літальним апаратом. Також слід враховувати, що на різній висоті потоки повітря можуть пересуватися з різною швидкість та у різних напрямках, тому обприскування з літака у вітряну погоду може виявитися доволі складним та неефективним заняттям;

– потенційний ризик забруднення довкілля – частина розпиленої речовини може не досягти свого першочергового призначення, а бути рознесеною вітром на сусідні ділянки та території. Окрім того через ту ж причину під загрозою забруднення можуть опинитися як територія самого господарства, так і прилеглі житлові території.

1.1.4 Обприскування через системи зрошення

Особливість цього методу полягає у відсутності залучення будь-якої техніки до процесу обприскування. Натомість речовина доставляється на ділянку системою труб та шлангів, після чого її розпилення відбувається через відповідні насадки чи клапани. Подібні системи обприскування можуть бути декількох видів: спрейові, дощувальні та крапельні. Головна їхня різниця полягає у способі розпилення речовини. Спрейові використовують насадки, які під час роботи створюють струмені речовини та поливають ними посіви або у певному кутовому діапазоні, або у повному колі. Дощувальні системи модернізують цей підхід, розпилюючи речовину вже не струменями, а дисперсною хмарою з маленьких крапель. Крапельні ж системи речовину не розпилюють, а доставлять у вигляді крапель в ті місця, де прокладено труби чи шланги.

Кожна з систем зрошення має як свої переваги, так і недоліки. Наприклад, спрейові системи мають більшу дальність покриття, за рахунок напрямленої дії,

але сильно програють в ефективності, оскільки дисперсія крапель в такому випадку є найменшою. Дощувальні ж навпаки мають непогану дисперсію, але програють в дальності. Крапельні системи взагалі є непридатними для обприскування як такого, а можуть бути використані більше для доставки рідких добрив прямо під корені рослин.

Утім усі ці системи мають деякі спільні переваги та недоліки у порівнянні з усіма іншими методами обприскування. Так наприклад, до переваг можна віднести:

– дешевизна – спорядження подібних систем є дешевшим за покупку спеціалізованої техніки;

– можливість використання у закритих приміщення – на рівні з ручним обприскування зрошувальні системи можуть бути використані у закритих приміщеннях таких як, наприклад, теплиці;

– дуальність використання – зрошувальні системи можуть надавати дві функції: обприскування пестицидами та, власне, саме зрошення;

– автономність – після спорядження системи участь людини у процесі обприскування обмежується увімкненням та вимкненням системи.

Але цей метод має і ряд суттєвих недоліків:

– важкість використання на великій площі – якщо мова йде про невеликі площі (до 1-2 га), то подібні системи є непоганим вибором, у випадках же з більшою площею починаються труднощі – в трубах тиск падає до критичного рівня, а тому на віддалених ділянках обприскування майже не відбуватиметься;

– складність встановлення – для ефективної роботи необхідно розрахувати розташування усіх точок розпилення з урахуванням падіння тиску з віддаленням від насосу, а потім розгорнути усю систему вручну;

– необхідність прибирання системи перед збором врожаю.

1.2 Обприскування за допомогою БПЛА, його переваги та недоліки

Обприскування з використанням БПЛА – передбачає використання різних видів БПЛА, які обладнані обприскувачами та здатні до дуже точного (а подекуди навіть точкового) розпилення хімічних речовин. Метод є дуже точним та ефективним, та може застосовуватися зокрема для оброблення важкодоступних ділянок та мінімізації витрат корисного навантаження (хімічних речовин).

Цей метод обприскування наразі набуває все більшої популярності в сільському господарстві. Метод є чимось середнім між тракторним обприскуванням та обприскуванням за допомогою літаків з точки зору способу доставки та розповсюдження хімічних речовин. Обприскування може проводитися різними типами БПЛА: літакового типу чи мультикоптерами, керованими чи автономними.

БПЛА літакового типу за своєю суттю є зменшеними версіями літаків, тому розпилення хімічних речовин з їхньою допомогою відбувається за схожим принципом як і з літаками. Але на практиці ніхто не використовує саме цей вид БПЛА у сільському господарстві через те, що він поділяє з літаками усі їхні недоліки, але при цьому не має тих самих переваг за рахунок зменшених розмірів. Тому найбільш розповсюдженим є використання мультикоптерів.

Мультикоптер (або ж просто коптер) – це вид БПЛА, який використовує довільну кількість несучих гвинтів для вільного пересування у просторі в усіх напрямках. Побудований мультикоптер за принципом гелікоптера, але замість одного несучого гвинта він має чотири і більше, та не має хвостового гвинта, оскільки для керуванням нахилу (а отже й напрямком руху) він використовує різницю потужностей окремих гвинтів. Зазвичай, мультикоптери використовують не менше чотирьох гвинтів та мають парну кількість цих гвинтів, що зроблено для більшої стабільності та керованості. Відповідно до кількості гвинтів мультикоптери мають варіацію назв, змінюючи префікс на відповідний множник, який позначає те чи інше число в грецькій мові: квадрокоптер (чотири гвинти), гексакоптер (шість гвинтів), октокоптер (вісім гвинтів) тощо.

БПЛА також поділяють за типом керування: керовані та автономні. Керовані БПЛА – це ті, які керуються людиною дистанційно за допомогою пульта, джойстика, панелі керування. Автономні ж БПЛА не потребують участі людини в процесі керування. Вони можуть переміщатися у просторі самостійно за заздалегідь встановленими координатами, або ж використовуючи алгоритми прокладання шляху чи штучний інтелект (ШІ). Автономні БПЛА також часто називають альтернативною назвою – дрони [2].

Хоч поняття дрона включає не тільки літальні апарати, а також й інші види автономних засобів, надалі в роботі буде використовуватися саме цей термін для позначення БПЛА типу мультикоптер.

Дрони для обприскування як правило більші за розмірами у порівнянні з дронами, що використовуються для спостереження. Такі дрони входять до категорії вантажних – таких, що спроєктовані на підняття вантажів значної ваги. Відрізняє ж їх від цілковито вантажних дронів те, що окрім резервуара для рідини вони мають обладнану систему розпилення цієї рідини, яка рівномірно розподілена по площі усього дрона.

Отже, обприскування за допомогою БПЛА являє собою застосування автономних мультикоптерів (дронів) для розпилення хімічних речовин над рослинами на певній висоті. Перевагами такого методу обприскування є:

– легка прохідність – дрон може легко дістатися ділянок, де великогабаритна техніка пройти не пройде;

– ретельний контроль над процесом – при використання дрона в оператора є можливість корегування його поведінки та маршруту в реальному часі;

– точність обприскування – дрон може проводити обприскування як окремих ділянок поля, так і точкові обприскування за рахунок великої маневреності та точної системи навігації;

– мінімізація витрат речовини – за рахунок чіткого контролю потоку речовини досягається оптимальна її витрата;

– ефективність – на відміну від усіх інших методів обприскування цей метод має найбільшу ефективність, оскільки він доставляє хімічні речовини точково, дозовано та з найбільш ефективною дисперсією крапель;

– автономність – від людини вимагається лише задати початкові параметри (та інколи ще маршруту) та завантажити дрон хімічною речовиною, після чого програма керування дроном проведе обприскування самостійно.

Хоч і може здатися, що обприскування за допомогою дронів не має недоліків, утім це так не є. Даний метод все ж має свої мінуси, такі як:

– мала вантажопідйомність – дрон, яким би великим він не був, не може оперувати тими ж об’ємами, що використовуються під тракторного обприскування, чи тим паче під час використання літаків. Вантажопідйомність одного дрона обмежується приблизно сімдесятьма кілограмами [3], що значно впливає на час обприскування за рахунок збільшення кількості необхідних дозаправок;

– обмежений заряд батареї – дрони працюють на електродвигунах, які живляться від змінних акумуляторних батарей, що є обмежувальним фактором для часу роботи одного дрона. Звісно, з дроном можна застосовувати не одну батарею, але в такому разі зростає і ціна. Окрім того для обробки дуже великих площ може не вистачити і декількох батарей, що змушує переривати процес обприскування на час їхньої зарядки;

– доступність – дрони є порівняно новітнім винаходом, а тому не завжди є вільно доступними на ринках непередових держав. Крім того ціна одного комплексу може сягати 80 тис. дол. США (приблизно 2,9 млн грн.) і більше [4], що цілком може перевищити ціни найдорожчих причіпних обприскувачів на ринку [5];

– швидкість обприскування – хоч у порівнянні з ручним обприскуванням цей метод і є швидшим, утім він все ж програє у швидкості тракторному обприскуванню та обприскуванню літальними апаратами, головне через необхідність заміни батареї або додаткової зарядки під час оброблення великих площ;

– залежність від погодних умов – хоч дрони все ж не є настільки залежними від погодних умов як великі літальні апарати, утім вони все ж піддаються впливу сильного вітру. Деякі великі моделі здатні витримувати пориви вітру до 7-8 м/с, утім найбільшою перешкодою є не вплив вітру на сам дрон, а скоріше його вплив на розпилену речовину, яка може бути віднесена від розрахованої точки нанесення.

Враховуючи перелічені переваги та недоліки, можна зробити проміжні висновки по тому, чи є обприскування за допомогою дронів кращим за традиційні методи. Але для початку необхідно визначитися з тим, що можна вважати кращим, та за якими критеріями це «краще» визначатиметься.

Однією з найпримітивніших моделей оцінки будь-якої роботи чи продукту є так званий «Залізний трикутник» (або «Недосяжний трикутник»). Концепція цієї моделі полягає в тому, що певну роботу чи проєкт можна охарактеризувати трьома факторами – швидкість, ціна, якість – при чому досягти ідеальних показників усіх трьох одночасно неможливо [6]. Відповідно до цієї моделі перед виробником товару чи послуги стоїть задача вибрати оптимальне співвідношення усіх трьох параметрів таким чином, щоб найбільше задовольнити попит.

Але подібне представлення є занадто спрощеним і може не відображати усіх особливостей того чи іншого продукту. Тому для аналізу методів обприскування буде краще застосувати більш просунуті методики. Так, наприклад, з цією метою можна використати пелюсткові діаграми, в яких вершини будуть позначати характеристики методу обприскування.

Для застосування подібних діаграм спочатку необхідно визначити, якими показниками можуть бути охарактеризовані ці методи. Перш за все, важливим показником кожного з методів обприскування є його вартість. Вартість може включати витрати на обладнання, обслуговування, інфраструктуру. Другим показником однозначно має бути ефективність, яка репрезентуватиме якість розпилення хімічної речовини. Третім показником буде швидкість виконання обприскування. Четвертим буде доступність, що визначається наявністю необхідного обладнання на ринку в потрібній кількості. П’ятим буде прохідність

– наскільки легко можна обробляти важкодоступні місця тим чи іншим методом. Шостим показником будуть ресурсні витрати, що включатимуть як витрати хімічної речовини та води, а також залученість людських ресурсів до роботи. І останнім сьомим показником буде залежність від погодних умов – наскільки вітряна погодна впливатиме на процес обприскування.

Кожний з показників буде вимірюватися в оцінці від 1 до 5, де 1 – найнижча оцінка, а 5 – найвища. Найвищу оцінку отримує той метод, який має найкращий показник серед усіх перелічених, найнижчу – той, який найгірший з усіх за вказаним показником.

В такому разі можна побудувати пелюсткову діаграму на основі визначених переваг та недоліків (рис. 1.1).

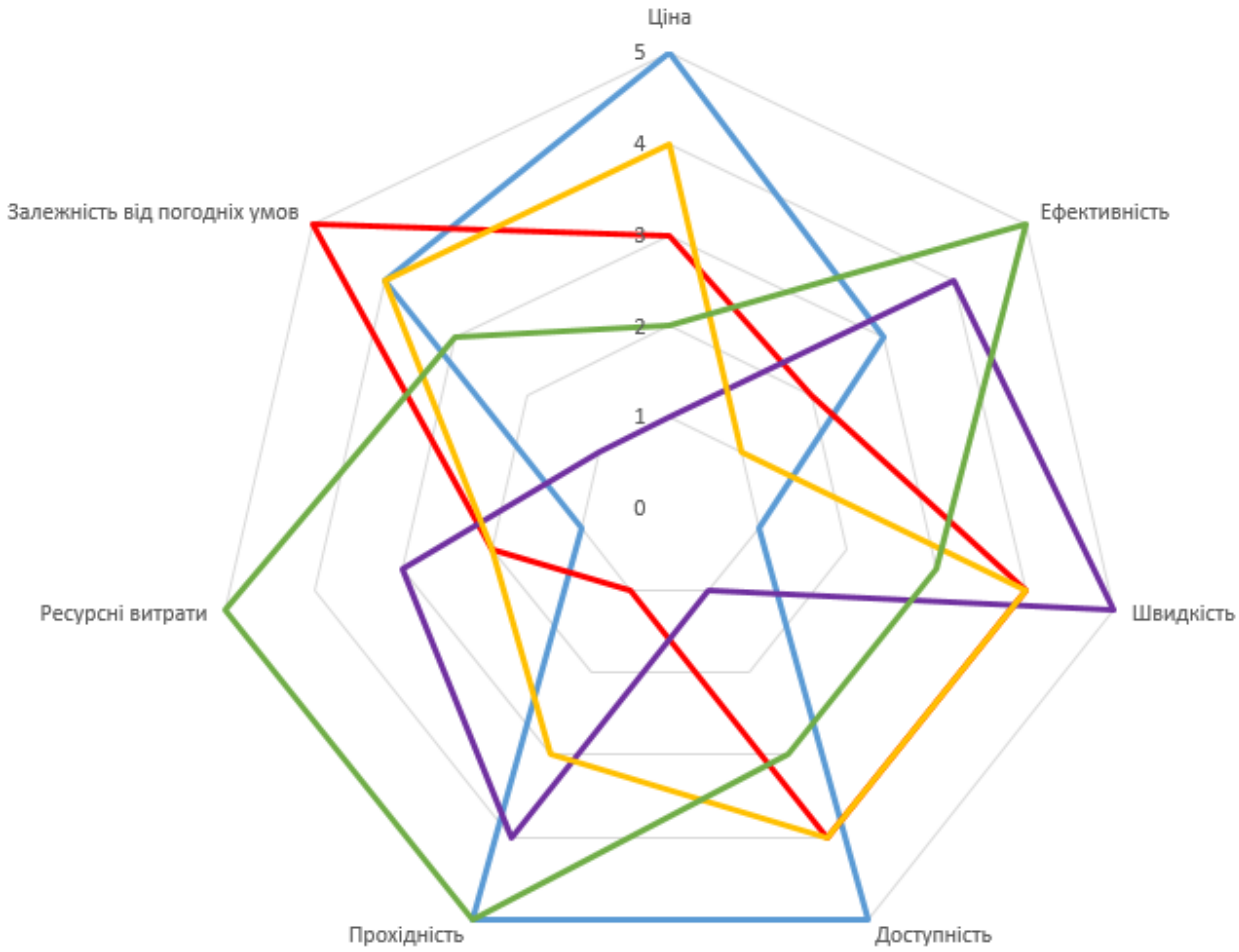

Рисунок 1.1 – Порівняльна діаграма методів обприскування (синій – ручне обприскування, червоний – тракторне, фіолетовий – літаком, жовтий – системами зрошування, зелений – дроном)

Як можна побачити з діаграми, кожен з методів має гарні показники в одній категорії, але погані в іншій. Як за принципом «Залізного трикутника» не можна досягти ідеальних показників в усіх трьох параметрах, так і в цьому випадку не вийде отримати максимум в усіх параметрах одночасно. Утім з діаграми видно, що деякі методи мають менші показники в середньому. Для кращого розуміння виведем середнє значення оцінки для кожного з методів: ручне обприскування – 3,43; тракторне обприскування – 3; обприскування за допомогою літальних апаратів – 2,71; обприскування з використанням систем зрошення – 3,14, обприскування за допомогою дронів – 3,71.

З цих оцінок стає зрозумілим, що в середньому обприскування за допомогою дронів випереджає усі інші методи. Це дає підстави стверджувати, що даний метод не просто так активно набирає популярність останнім часом.

Утім стверджувати, що цей метод є об’єктивно кращим за інші все ще не можна, оскільки подібне порівняння базується не на емпіричних знаннях, а лише на теоретичному узагальненому погляді. Тому на основі цього порівняння можна лише висунути теорію, що обприскування за допомогою дронів є об’єктивно кращим рішенням з точки зору ефективності та витрат ресурсів.

Але на користь цієї теорії говорять і деякі дослідження. Так, наприклад, у дослідженні [7] порівнювали ефективність обприскування за допомогою дронів та конвенційних обприскувачів. У дослідженні зроблено висновок, що купівля дронів та усього супровідного обладнання до них в середньому обходиться дорожче, ніж придбання традиційних обприскувачів. Але за рахунок того, що дрони потребують приблизно на 38% менше витрат на обслуговування, вони мають майже вдвічі більшу економічну інверсію, що означає те, що дрони окупаються швидше. Також в дослідженні порівнювали ефективність роботи цих двох видів обприскувачів, і виявилося, що коли конвенційні обприскувачи мали ефективність в середньому 1,6 га/год, то ефективність дронів в цей час була від 5 до 7 га/год.

В іншому дослідженні [8] проводилось вимірювання впливу обприскування за допомогою дрона на посіви рису в Індонезії. У порівнянні з ручним

обприскуванням дрон показав більшу ефективність. Також у цьому дослідженні зазначається, що дрон виконав роботу значно швидше за ручний метод, і що при використанні дрона негативний вплив пестицидів на здоров’я працівників плантацій вагомо зменшується.

Також у схожому дослідженні [9] було проведено аналіз ефективності застосування дрона для обприскування рисових полів. Результатом використання дрона замість ручної обробки стало зменшення кількості залученого персоналу приблизно на 37,5%, зменшення часу обробки майже в чотири рази, підвищення врожайності полів на 9,1% (з 13,3% до 22,4%, тобто ефективність зросла майже на 70%).

Отже, виходячи з розглянутих емпіричних досліджень можна зробити висновок, що запропонована теорія виправдалася і використання дрона дійсно збільшує як швидкість обробки, так і врожайність оброблених ділянок у порівнянні з традиційними методами обприскування. Тому поширення практики використання дронів для цих цілей є перспективним завданням, яке може призвести до загального покращення виробництва сільськогосподарської продукції та зменшення кількості необхідних для виробництва людських ресурсів.

1.3 Визначення проблеми та постановка задачі

Кожен з перелічених вище методів використовується в тій чи іншій мірі в усьому світі. Утім деякі з них є більш популярними, за інші в силу таких факторів як: розміри сільських господарств, типи вирощуваних культур, доступ до відповідних технологій та техніки, розмір бюджету. Так наприклад, у розвинених країнах, де є вільний доступ до найновіших технологічних засобів, все більшої популярності набувають самохідні обприскувачі та дрони через їхню точність та ефективність. У той же час у країнах, що розвиваються, можуть застосовуватися більш традиційні методи обприскування, такі як тракторне або ручне обприскування.

В Україні ж найпопулярнішим методом є тракторне обприскування. Поширеність саме цього типу зумовлена рядом факторів:

– великі та просторі площі полів, на яких легко може оперувати великогабаритна техніка;

– доступність використання певного виду техніки – після розпаду СРСР сільськогосподарським підприємствам залишився в спадок немалий автопарк сільськогосподарської техніки, в якому переважну частину займали саме трактори, що значною мірою вплинуло на подальший вектор розвитку цього напрямку;

– нерентабельність використання авіаційної техніки – після здобуття Україною незалежності галузь малої та середньої авіації почала вимирати, оскільки не була пристосована до функціонування без державного регулювання та субсидіювання. Оскільки у новоствореної держави не було коштів, які б можна було виділити на цей сектор, аеродроми місцевого рівня почали активно згортати свою діяльність, а наявний авіапарк почали продавати, або здавати на металобрухт. Таким чином на даний час в Україні відсутня необхідна інфраструктура для використання малої авіації для потреб обприскування полів.

Важливим критерієм обрання методу обприскування є співставлення ряду їх якісних та кількісних характеристик, які визначають їхню ефективність. До кількісних можна віднести:

– площа обробки – кількість землі, яку може обробити тракторний метод обприскування за один робочий день або годину;

– витрати палива – визначається кількістю пального, яке використовується трактором для обприскування певної площі поля;

– витрати хімічних матеріалів – кількість хімічних речовин, яка розпилюється на поле для захисту рослин. Важливо мінімізувати витрати хімічних матеріалів для збереження витрат та запобігання забрудненню навколишнього середовища;

– швидкість роботи – вказує на те, як швидко може трактор виконувати обприскування. Висока швидкість роботи може підвищити продуктивність;

– ширина обприскування – розмах розпилювачів або штанг, що використовуються для обприскування. Ширше покриття дозволяє обробити більшу площу за один прохід.

До якісних характеристик же можна віднести:

– рівномірність покриття – важливо, щоб обприскувачі розподіляли хімічні речовини рівномірно на рослини. Нерівномірне покриття може призвести до недообробки або переобробки ділянок поля;

– точність дозування – для ефективного контролю шкідників і бур'янів важливо, щоб обприскувачі точно дозували хімічні матеріали. Надмірне чи недостатнє дозування може бути неефективним;

– можливість налаштування – якість тракторного методу обприскування також визначається його можливістю налаштування для різних типів культур, площ та умов. Гнучкість властивостей дозволяє підвищити ефективність використання;

– тривалість служби та надійність – надійність та тривалість служби техніки також важливі для забезпечення безперебійної роботи протягом сезону.

Враховуючи усі перелічені вище характеристики, можна заключити, що тракторний метод в Україні набув найбільшої популярності не просто так. Значну роль відіграла не лише доступність технічних засобів, а й оптимальні характеристики цього методу в порівнянні з усіма іншими.

В теперішній же час ситуація змінилася суттєво. Трактори вже не є найбільш ефективним методом обприскування, оскільки з’явився новий вид технічних засобів – дрони.

Насправді, дрони почали використовувати ще в першій половині XX ст. для фотографування місцевості, створення мап сільськогосподарський полів, моніторингу росту посівів та визначення необхідності використання пестицидів. Але вже на початку 2000-х років дрони почали використовувати масово для виконання різноманітних задач [10]. В Україні, на противагу більш технологічно розвиненим країнам, використання дронів у сільському господарстві почалося відносно нещодавно. З розвитком технологій дрони стають все дедалі дешевшими

та доступнішими для все більшої кількості підприємств. Таким чином вже на даний момент в Україні існують підприємства, які застосовують дрони для робіт на полі. Втім відсоток їх використання все ще є доволі низьким у порівнянні з іншими розвиненими країнами.

Використання дронів для обприскування є одним з найпопулярніших методів їх застосування. Використання їх у цьому напрямку є не тільки економічно більш вигідним, а й може зберегти життя та здоров'я працівників сільськогосподарських підприємств. Постійна відкритість працівників аграріїв до хімічних речовин під час їхньої роботи у полі може призводити до великого спектру захворювань – починаючи від хронічних захворювань, закінчуючи раком. Тому дуже важливою є безпека працівників під час процесу обприскування. Дрони ж можуть суттєво, якщо навіть не повністю, нівелювати подібну небезпеку. Оскільки для їх використання присутність людини поряд із зоною, що обробляється, не є необхідною, це може значно знизити ризики для здоров'я працівників.

Але для повноцінного та комфортного використання дронів для потреб обприскування недостатньо просто мати необхідний дрон. Його ще необхідно запрограмувати на виконання тих чи інших завдань. Завдання можуть відрізняти за рядом факторів: вид дрона, розмір поля, висоти місцевості, наявність перешкод, погодні умови, тип посівів та хімічного матеріалу для обприскування тощо. Окрім того, для застосування дронів в автономному режимі необхідно задати йому маршрут, за яким він буде здійснювати обприскування. Робити все це вручну для кожного дрона при кожному новому запуску не є ефективним. Саме тому існують відповідні інформаційні системи, що надають зручний і зрозумілий інтерфейс та забезпечують увесь потрібний функціонал налаштування та контролю обприскування за допомогою дронів.

Тому створення інформаційної системи, яка б мала змогу налаштування обприскування полів із залученням різних дронів, є доволі перспективною задачею. Саме розробка подібної системи і є головним завданням в даній роботі.

Висновки до розділу 1

Під час написання цього розділу було проаналізовано методи обприскування сільськогосподарських полів, які наразі є найпопулярнішими у світі. Серед них визначено умовно традиційні (ручне обприскування, тракторне обприскування, обприскування за допомогою літальних апаратів та систем зрошення) та умовно сучасний (обприскування за допомогою дронів). В ході аналізу було досліджено сутність кожного з методів, їхню значущість у сучасному сільському господарстві, а також їхні переваги та недоліки.

На основі визначених переваг та недоліків було проведено теоретичний порівняльний аналіз зазначених методів, результатом якого стала теорія, що метод обприскування за допомогою дронів є найбільш ефективним у порівнянні з іншими методами, які розглядалися. Для підтвердження цієї теорії було проведено огляд наукової літератури та передових досліджень у галузі застосування дронів у сільському господарстві. Розглянуті емпіричні дослідження загалом підтверджують висунуту в цій роботі теорію – обприскування за допомогою дронів в середньому має вищі показники ефективності, й окрім того воно є швидшим за деякі методи та значно безпечнішим для робітників сільськогосподарських підприємств.

Також в цьому розділі розглянуто становище сільського господарства в Україні, а також те, які методи обприскування є найбільш популярними. Визначено, що найбільш популярним є метод тракторного обприскування. Виходячи з отриманого результату дослідження, сформовано задачу цієї роботи – розроблення інформаційної системи, яка б дала змогу налаштування обприскування полів за допомогою різних дронів.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Основні види продукції з обприскування за допомогою БПЛА на ринку

Для того щоб визначити функціональні вимоги до майбутньої системи, необхідно спочатку проаналізувати вже наявні рішення, які присутні на ринку. Нажаль, подібних технологій дуже часто немає у загальному доступі, оскільки підприємства, які надають послуги з обприскування за допомогою дронів, дивляться на них як на корпоративну таємницю. Саме тому без особистого контакту з системами цих компаній процес аналізу може бути здійснений здебільшого лише за описом послуг, які надає та чи інша компанія.

2.1.1 Drone Survey Services

Drone Survey Services є провідною компанією у Великій Британії, яка надає послуги з обприскування сільськогосподарських культур безпілотними літальними апаратами, сумісними з HSE. Компанія використовує сільськогосподарські дрони для ефективного та сталого обприскування культур з повітря [11].

Компанія на своєму сайті зазначає, що може застосовувати різні способи обприскування посівів (рис. 2.1):

– загальне обприскування – вид обприскування, під час якого покривається уся площа поля (найшвидший та найдешевший спосіб із запропонованих);

– середній рівень покриття – обприскуються відповідні зони поля (середній за витратами спосіб);

– точковий спосіб – обприскується багато невеликих зон (найдовший і найдорожчий спосіб, але з найбільшим показником ефективності).

Також компанія пропонує використання груп дронів для швидшого виконання завдань, а також обприскування лише тих ділянок, що цього дійсно потребують, що має зменшити розхід ресурсів (рис. 2.2).

Рисунок 2.1 – Способи обприскування від компанії Drone Survey Services

Рисунок 2.2 – Додаткові можливості, які пропонує компанія Drone Survey Services

Також на сайті компанії зазначається, що їхні дрони оснащені передовими сенсорами, які здатні до точного позиціонування дронів у просторі, автономного визначення необхідних ділянок поля, що потребують обприскування, а також вони можуть визначати точну кількість речовини, яку необхідно розпилити для досягнення бажаного ефекту (рис. 2.3).

Серед переваг цієї компанії можна виділити:

– можливість вибору між кількома способами обприскування;

– використання декількох дронів для спільної роботи над однією ділянкою;

– використання сучасних датчиків позиціонування та спостереження.

До недоліків можна віднести:

– відсутність коригування поведінки дронів в залежності від погодних умов;

– відсутність обчислення оптимального маршруту обприскування;

– відсутність можливості збереження інформації про робочі зони та налаштувань обприскувань.

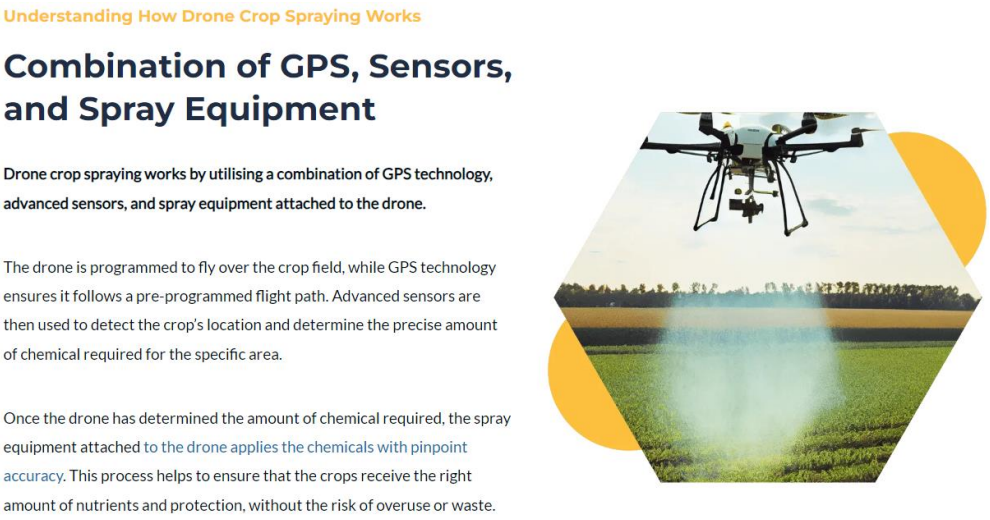

Рисунок 2.3 – Технології, що використовує компанія Drone Survey Services

2.1.2 Auto Spray Systems

Ще одна компанія, яка пропонує свої послуги з автономного обприскування у Великобританії. Ця компанія пропонує послуги з обприскування з використанням дронів лінійки XAG (рис. 2.4). Окрім повітряних компанія також має у своєму розпорядженні наземні обприскувальні дрони. Як і в попередньому випадку, ця компанія не надає чітких функціональних можливостей своєї системи, утім має відео, які демонструють процес роботи їхніх дронів (рис. 2.5). З цих відео, наприклад, можна виділити одну з можливостей їхньої системи – а саме планування траєкторії обприскування на ділянках із різною висотою.

Отже, з цього можна сформувати наступні переваги їхньої системи:

– можливість використання дронів як повітряного, так і наземного типів;

– можливість використання дронів на ділянках із різною висотою.

До недоліків же відноситься:

– застосування дронів лише одної конкретної лінійки;

– відсутність коригування поведінки дронів в залежності від погодних умов.

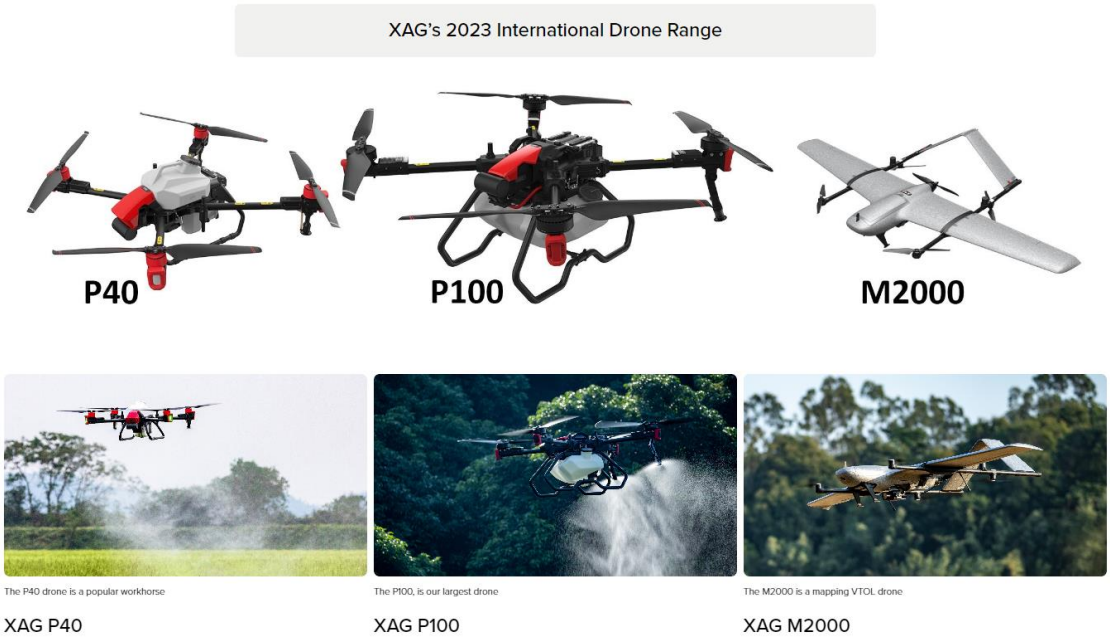

Рисунок 2.4 – Парк повітряних дронів компанії Auto Spray Systems

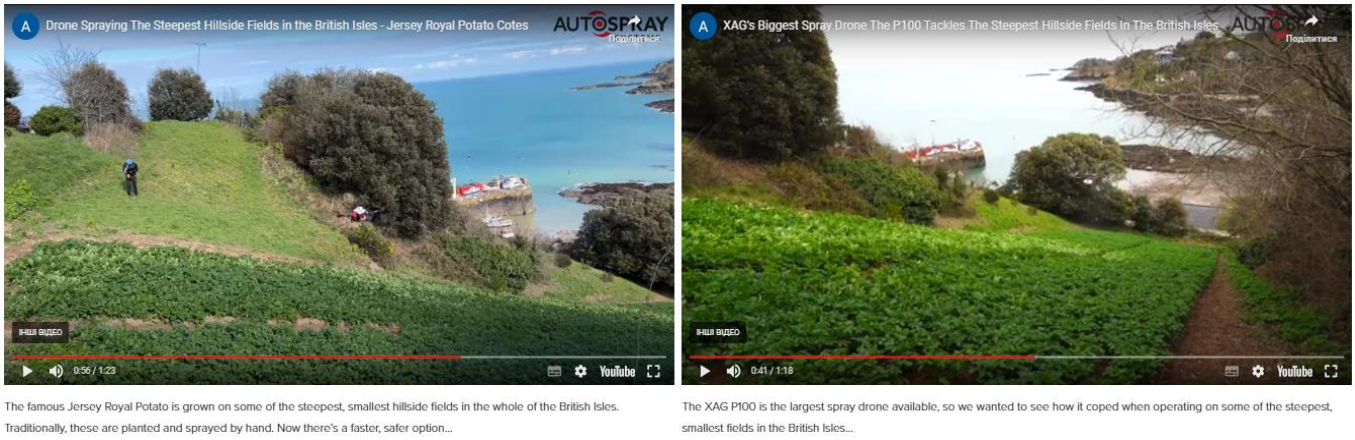

Рисунок 2.5 – Відео-демонстрація роботи дронів компанії Auto Spray Systems

2.1.3 Hylio AgroSol

AgroSol – це флагманське програмне забезпечення компанії Hylio для наземного керування. Як стверджує компанія, AgroSol легка для навчання та оптимізована для сільськогосподарських робіт. Система пропонує легку можливість скласти карту, спланувати та виконати обробку культур.

Однією зі своїх переваг компанія зазначає можливість легкого управління без необхідності великої кількості налаштувань. Все що треба це вказати завдання, натиснути «Злетіти», а потім спостерігати, як дрони роблять решту.

Щойно вони закінчать свою роботу або закінчиться корисне навантаження, вони автоматично повернуться додому для наступного завдання.

На противагу попереднім прикладам, ця компанія на своєму сайті має демонстрацію роботи із системою. Система надає користувачу широкий функціонал:

– керування зонами роботи (планування траєкторії, налаштування параметрів роботи, завантаження/вивантаження цієї інформації з/до хмари) (рис. 2.6);

– керування завданнями та можливість призначення на завдання наявних дронів (рис. 2.7);

– підтримка обслуговування дронів (система моніторингу їхнього стану, детальні налаштування кожного окремого дрона);

– формування звітності про проведені роботи (перегляд та вивантаження звітів про виконані завдання з детальним їхнім описом та результатом) (рис. 2.8).

Все вище зазначене можна віднести до переваг цієї системи. З недоліків можна виділити хіба що відсутність вебверсії клієнта системи. Щоб керування роботою можна було здійснювати на місці, а не тільки з кабінету сільськогосподарського підприємства, треба мати можливість доступу до системи також з мобільних пристроїв (смартфонів або планшетів).

Рисунок 2.6 – Демонстрація можливостей системи Hylio AgroSol у роботі із зонами

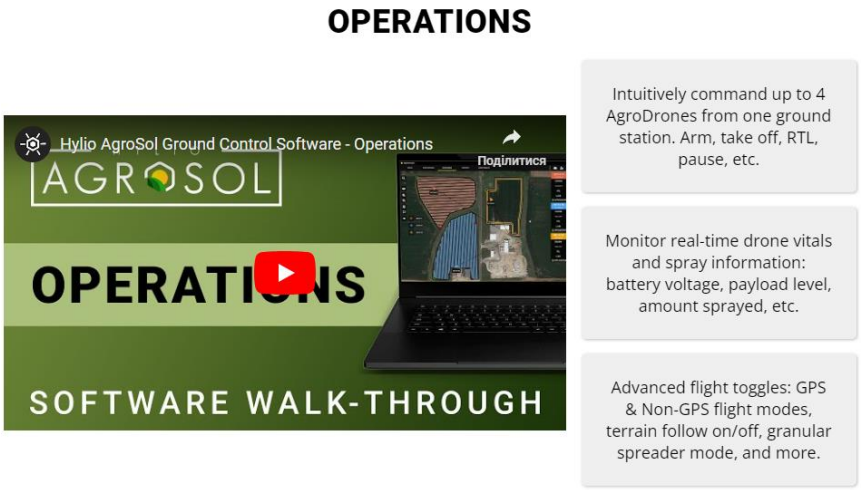

Рисунок 2.7 – Демонстрація можливостей системи Hylio AgroSol у роботі з операціями

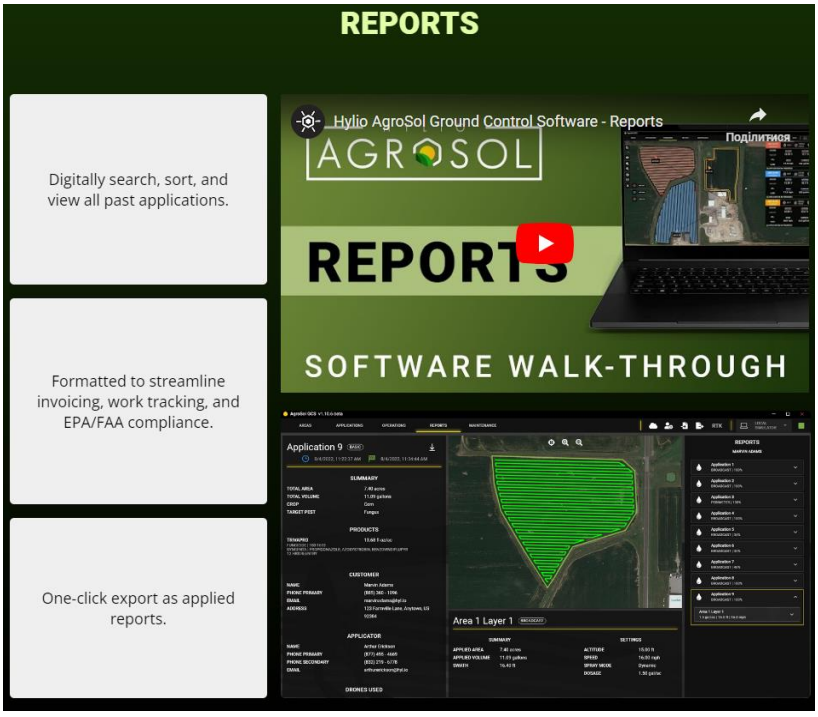

Рисунок 2.8 – Демонстрація можливостей системи Hylio AgroSol у роботі зі звітами

2.1.4 Argemo

Agremo є доступною сільськогосподарською платформою та програмним забезпеченням для аналізу даних, яке використовує штучний інтелект для

виконання аналізу аерофотознімків. Мета системи – забезпечити користувачам точну інформацію про їхні поля, використовуючи актуальні дані та аналітику протягом вегетаційного періоду. Платформа дозволяє агрономам, операторам дронів, фермерам та іншим сільськогосподарським професіоналам ефективно планувати, контролювати та аналізувати сільськогосподарську діяльність, щоб підвищити продуктивність культур, збільшити врожайність і знизити витрати виробництва. Платформа також призначена для консультантів, які мають намір покращити продуктивність сільського господарства, подвоїти свою продуктивність і збільшити прибуток.

Хоча дана система призначена не для обприскування, а для задачі моніторингу, вона все ж може бути корисною для аналізу користувацького інтерфейсу та потенційного функціоналу.

Отже, Agremo надає користувачам доволі простий інтерфейс, який складається з карти, що займає більшу частину вікна, та допоміжних меню, які можуть бути як спливаючими формами, так і розташованими з одного з боків вікна. Так, наприклад, процес початку аналізу показано на рисунках 2.9-11. Як можна побачити з цього прикладу, на карті розташовуються області полів, які можна виділяти для того, щоб почати аналіз, увівши необхідні параметри до спливаючи форми.

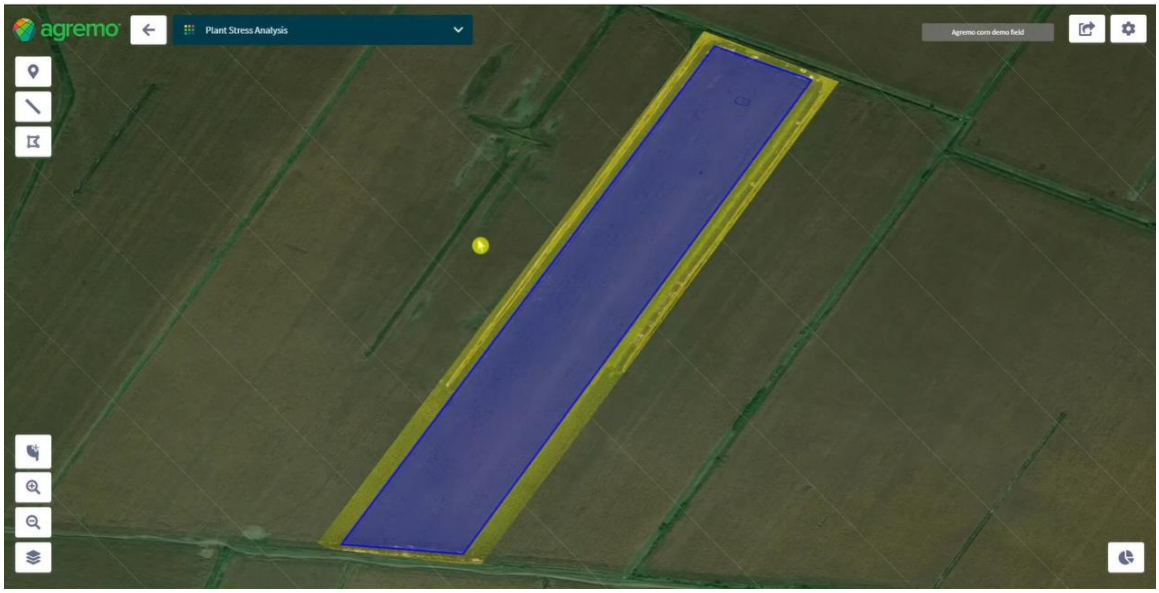

Рисунок 2.9 – Загальний вигляд інтерфейсу системи Agremo

Рисунок 2.10 – Виділення області для аналізу в системі Agremo

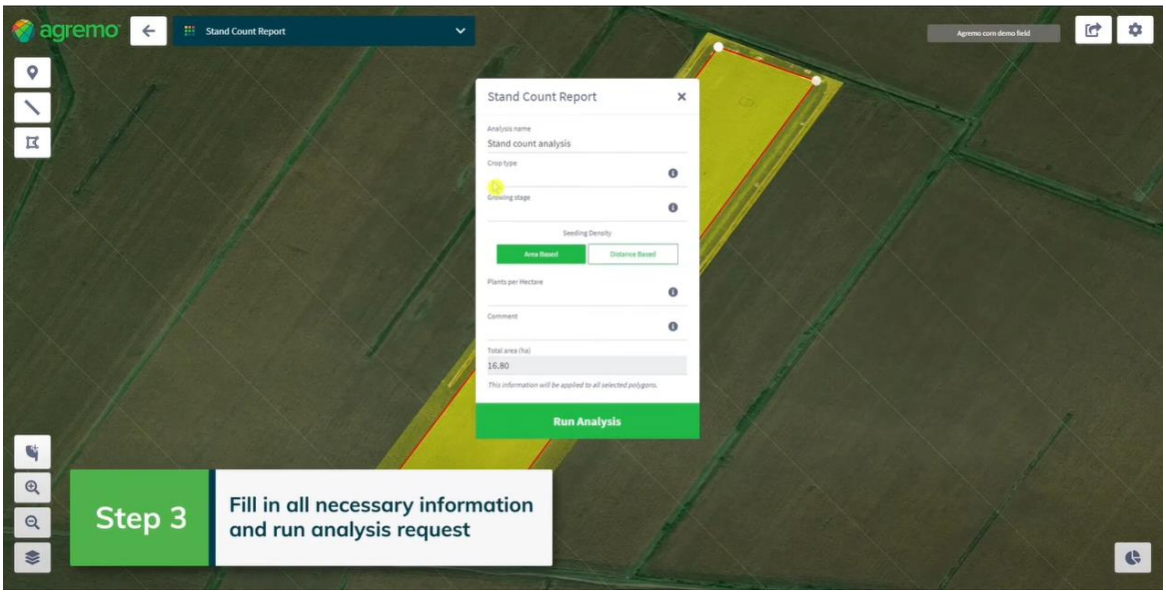

Рисунок 2.11 – Форма початку аналізу в системі Agremo

Також система дозволяє наносити на карту нові області. Процес додавання нової області проілюстрована на рисунках 2.12-14. Таким чином система надає користувачам бокове меню з вибором можливих дій, таких як: вибір функціоналу, нанесення об’єктів на карту чи можливість поділитися картою або провести її вивантаження (рис. 2.12). На рисунку 2.13 можна побачити процес нанесення нової області на карту, вибравши спочатку відповідну функцію в меню.

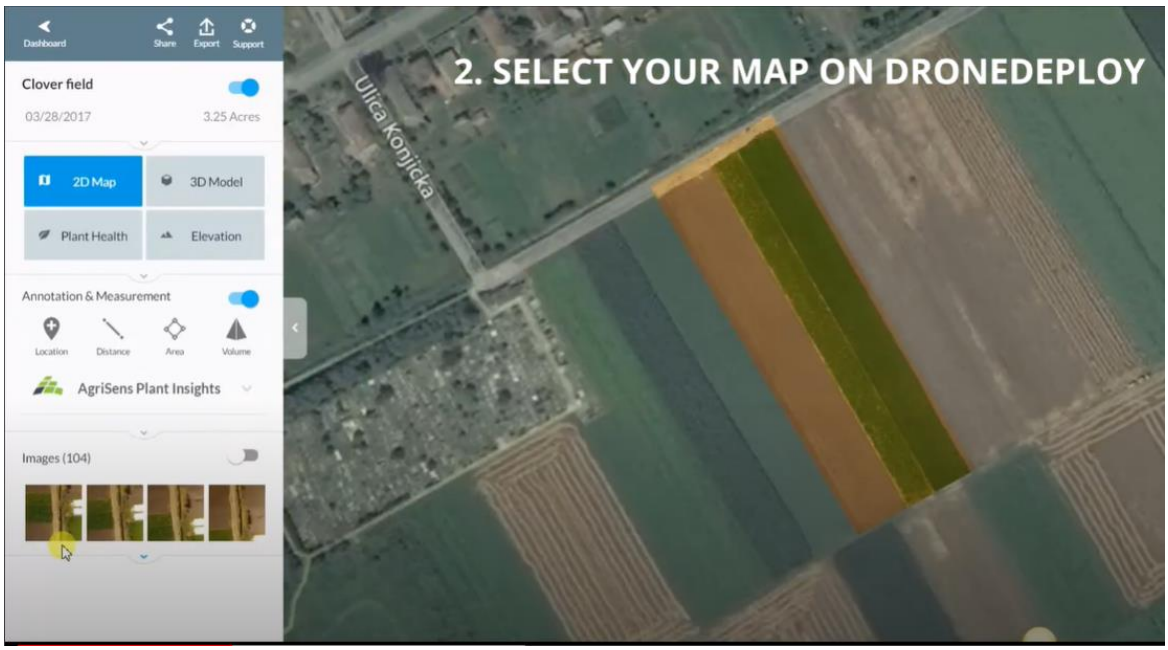

Рисунок 2.12 – Вибір карти для додавання області системи Agremo

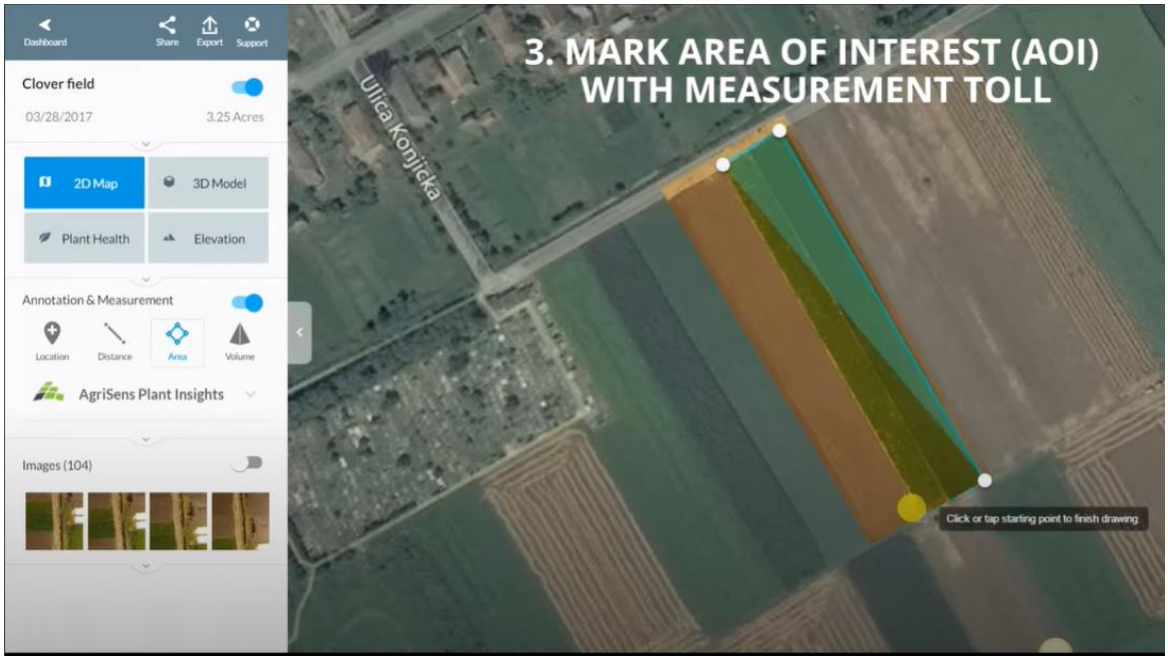

Рисунок 2.13 – Процес додавання нової області в системі Agremo

Після додавання області її можна виділити й, увівши усі необхідні параметри до форми, почати аналіз (рис. 2.14). Після цього у вкладці «Результати» можна буде спостерігати процес формування результатів аналізів, а також після їх завершення відкривати сформовані звіти.

Рисунок 2.14 – Внесення параметрів для початку аналізу, використовуючи новододану область, в системі Agremo

2.2 Формування вимог до системи

Оскільки метою даної роботи є розробка інформаційної системи з підтримки обприскування сільськогосподарських полів за допомогою БПЛА, необхідним кроком є визначення функціональних вимог до цієї системи, які б враховували усі переваги та недоліки розглянутих рішень. Таким чином для того, щоб система, яка розробляється, мала визначені переваги та за можливості не мала визначених недоліків, сформовано наступні функціональні вимоги:

– система повинна мати карту для керування робочими областями (РО) та точками завантаження (ТЗ);

– система повинна надавати користувачам можливість створювати нові РО і ТЗ, додаючи їх на карту, з подальшою можливістю їх збереження до бази даних (БД);

– система повинна надавати користувачам можливість створення, редагування та видалення робочих планів (РП), які використовуватимуться для програмування обприскування;

– РП повинні містити дані про: обрані РО та ТЗ (по одній на РП); дрон, який виконуватиме роботу; параметри швидкості дрона, ширини смуги обприскування та об’ємної витрати рідини;

– під час створення РП система повинна автоматично обраховувати оптимальну траєкторію обходу покриття (ТОП) для обраної РО з урахуванням обраної ТЗ;

– система повинна надавати користувачам можливість додавання, редагування та видалення моделей дронів, які використовуватимуться в роботі;

– система повинна надавати користувачам можливість додавання, редагування та видалення дронів, які використовуватимуться в роботі та базуватимуться на доданих моделях дронів;

– система повинна надавати користувачам можливість запуску створених РП з можливістю відслідковування процесу обробки в режимі реального часу.

Висновки до розділу 2

Під час написання цього розділу було проаналізовано існуючі рішення, визначено їх основні переваги та недоліки. На розгляд було представлено 4 продукти, але не всі з них повністю відповідають тому виду системи, розроблення якої було поставлення за мету в цій роботі. Так 2 з розглянутих продукти лише надають сервісні послуги з обприскування без доступу до власної системи. Четвертий же розглянутий приклад орієнтований не на обприскування, а на моніторинг посівів.

Утім навіть з таким переліком розглянутих рішень було визначено основні функції, які має забезпечувати система, що розробляється. Серед найважливіших це керування РО та ТЗ (зокрема і з візуальним відображенням), керування дронами та їх моделями, керування РП та проведення самого обприскування з можливістю моніторингу прогресу в режимі реального часу.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

3.1 Загальна архітектура системи

Загалом існує багато різних способів побудови архітектури інформаційних систем. Найбільш популярними та найбільш уживаними можна вважати наступні:

– монолітна архітектура;

– клієнт-серверна архітектура;

– мікросервісна архітектура;

– архітектура, керована подіями (event-driven architecture, EDA);

– багаторівнева архітектура;

– однорангова архітектура (peer-to-peer, P2P);

– архітектура обміну повідомленнями, керована подіями;

– гібридні архітектури.

3.1.1 Монолітна архітектура

Монолітна архітектура – це традиційна модель архітектури програмного забезпечення, яка побудована як єдине ціле, як самодостатня та незалежна від інших програм одиниця. Термін «монолітний» часто асоціюється з величезним і масивним, що відображає суть монолітної архітектури для розробки програмного забезпечення. Ця архітектура включає велику обчислювальну мережу з єдиною кодовою базою (single code base), яка об’єднує всі бізнес-завдання (рис. 3.1). В цьому контексті єдина кодова база означає використання одного репозиторію з вихідним кодом, що дозволяє розробнику збудувати застосунок один раз і запустити його на різних платформах одночасно без необхідності будувати для кожної з них окремо.

Зміни в такому програмному продукті потребують оновлення всього стеку, доступу до кодової бази та створення та реалізації оновленої версії інтерфейсу сервісу. Це призводить до обмежень і тривалих оновлень. У разі розробки з

використанням монолітної архітектури основною перевагою є швидкість розробки за рахунок простоти застосунку на основі єдиного коду.

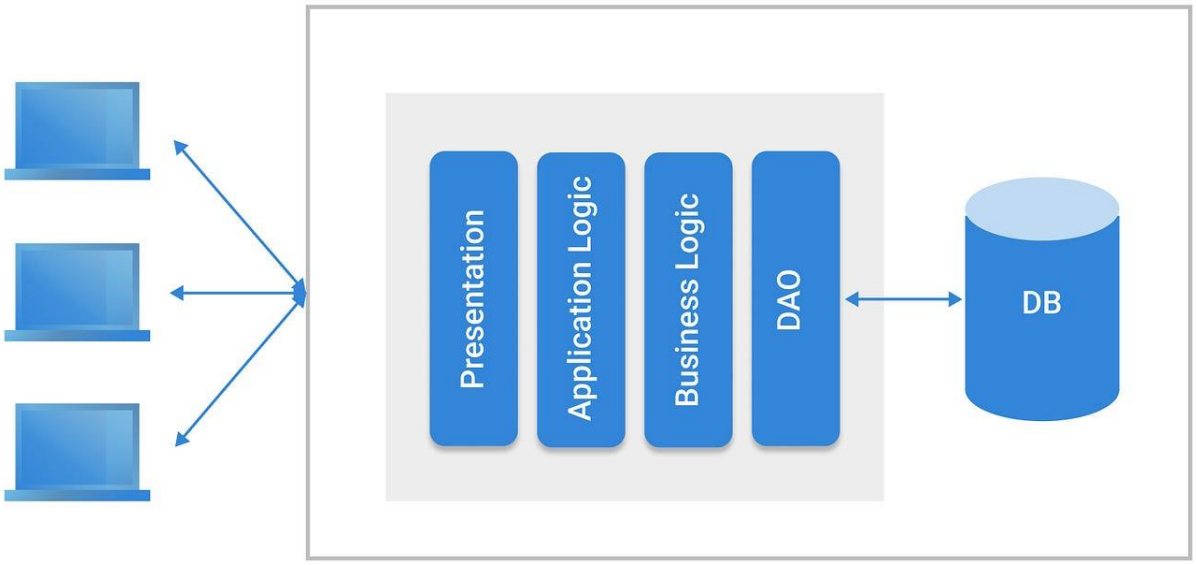

Рисунок 3.1 – Ілюстрація монолітної архітектури [12]

До переваг монолітної архітектури можна віднести:

– простота розгортання – за рахунок наявності лише одного виконуваного файл або каталогу спрощується процес розгортання;

– простота розробки – розробка застосунку, який має єдину кодову базу, є значно простішою;

– продуктивність – в централізованій кодовій базі єдиний API може надавати такі ж функції, що й багато різних мікросервісних API;

– спрощене тестування – наскрізне тестування може виконуватися швидше завдяки централізованому характеру монолітної програми;

– простота налагодження – розташування усього коду в одному місці полегшує пошук і вирішення проблем.

До недоліків монолітної архітектури можна віднести:

– повільніша швидкість розробки – на противагу простоті розробки доводиться жертвувати її швидкістю;

– обмежена масштабованість – через зосередження усього коду в одному місці підвищується його пов’язаність та взаємозалежність, через що масштабування окремих компонентів майже неможливе;

– хитка надійність – через помилку в одному модулі може відключитися увесь застосунок;

– важке впровадження технологій – будь-які зміни впливають на весь застосунок, що робить їх дорогими і трудомісткими;

– відсутність гнучкості – застосунки, розроблені за монолітною моделлю, обмежені технологіями, на яких вони були вперше побудовані;

– можливі ускладнення під час розгортання – невеликі зміни вимагають передислокації всього моноліту.

3.1.2 Мікросервісна архітектура

Архітектура мікросервісів (також відома просто як мікросервіси) – це метод архітектури, заснований на ряді окремо розгорнутих сервісів. Ці сервіси мають власну бізнес-логіку та відповідні бази даних із певним призначенням (рис. 3.2). Оновлення, тестування, розгортання та масштабування виконуються в межах кожного окремого сервісу. Мікросервіси розбивають основні бізнес-завдання на окремі автономні кодові бази. Такий підхід не зменшує загальну складність, але робить її більш зрозумілою та контрольованою завдяки розподілу завдань на менші незалежні процеси, які працюють автономно та сприяють загальному функціонуванню системи.

Використання мікросервісної архітектури набуває все більшої популярності, оскільки цей підхід має численні переваги. Серед них:

– принцип єдиної відповідальності – мікросервіси дотримуються принципу єдиної відповідальності, що означає, що кожна служба має лише одну відповідальність, що полегшує їх розробку та підтримку, а також допомагає зменшити ризик помилок під час внесення змін до кодової бази;

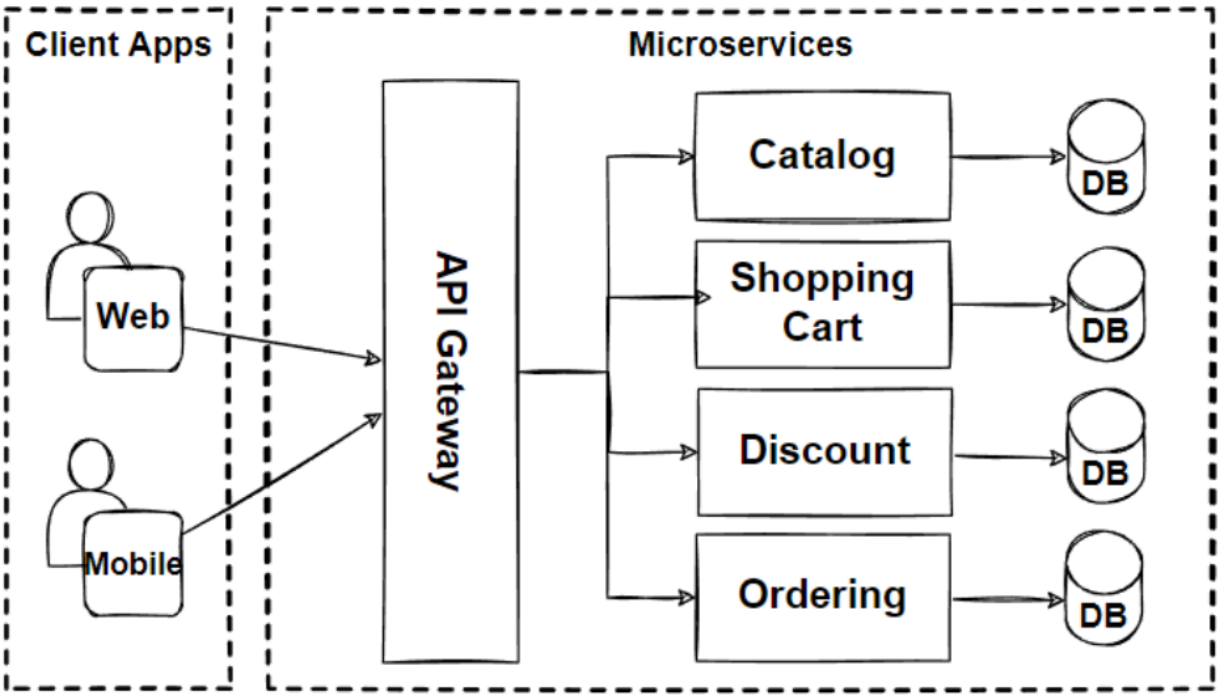

Рисунок 3.2 – Ілюстрація мікросервісної архітектури [13]

– масштабованість – кожен мікросервіс виконує певне завдання, і якщо попит на це завдання змінюється, можна змінити потужність лише відповідного сервера мікросервісу;

– швидкий час розгортання – завдяки невеликому розміру кожного мікросервісу вони швидше розгортаються порівняно з монолітними програмами;

– покращена відмовостійкість – кожен мікросервіс є окремим компонентом, тому його збій не впливає на інші частини системи. Це також дозволяє розгортати механізми безпеки для забезпечення відмовостійкості;

– менша вартість масштабування – розбиття програм на менші компоненти полегшує керування мікросервісами та їх масштабування. Кожен мікросервіс можна масштабувати індивідуально, що зменшує витрати на масштабування та робить їх більш рентабельними;

– краща продуктивність – мікросервіси дозволяють оптимізувати кожен компонент для конкретного завдання, що підвищує продуктивність, скорочує час завантаження та покращує взаємодію з користувачем. Це також допомагає зменшити витрати компаній.

Хоча покращена масштабованість і гнучкість роблять мікросервіси популярними, є кілька аспектів, про які розробники повинні знати. Ці недоліки включають:

– складність координації – взаємодія між різними мікросервісами ускладнюється їх розподіленим характером. Забезпечення надійності каналів зв'язку між ними стає ключовим завданням, оскільки багато таких комунікацій відбувається через мережу;

– збільшення часу розробки – мікросервіси, будучи більш складними, вимагають більше часу на розробку через необхідність координації між ними. Але оптимізувати час розробки можна за допомогою структурованого та організованого коду, а також за допомогою таких інструментів, як контейнери та безсерверні обчислення.;

– налагодження нової інфраструктури – разом із впровадженням мікросервісів необхідно створити додаткові компоненти, такі як черги повідомлень, конвеєр розгортання, шлюз API тощо;

– вища вартість впровадження – нова інфраструктура та збільшений час розробки призводять до підвищення вартості впровадження мікросервісів.

3.1.3 Клієнт-серверна архітектура

Клієнт-серверна архітектура відноситься до систем, які розміщують, надають та керують більшістю ресурсів і послуг, запитуваних клієнтом. У цьому форматі всі запити та послуги передаються мережею, тому цю архітектуру також називають мережевою або клієнт-серверною моделлю.

Клієнт-серверний застосунок – це мережева програма, яка розподіляє завдання та навантаження між клієнтами та серверами, які можуть знаходитися в одній системі або бути частиною комп’ютерної мережі.

Як правило, клієнт-серверна архітектура включає кілька користувальницьких робочих станцій, які підключені до єдиного центрального сервера через мережу Інтернет або інший вид мережі (рис. 3.3). Клієнт ініціює

запит на дані, сервер же в свою чергу приймає та обробляє цей запит, після чого надсилає пакети даних назад користувачеві з відповідними відповідями.

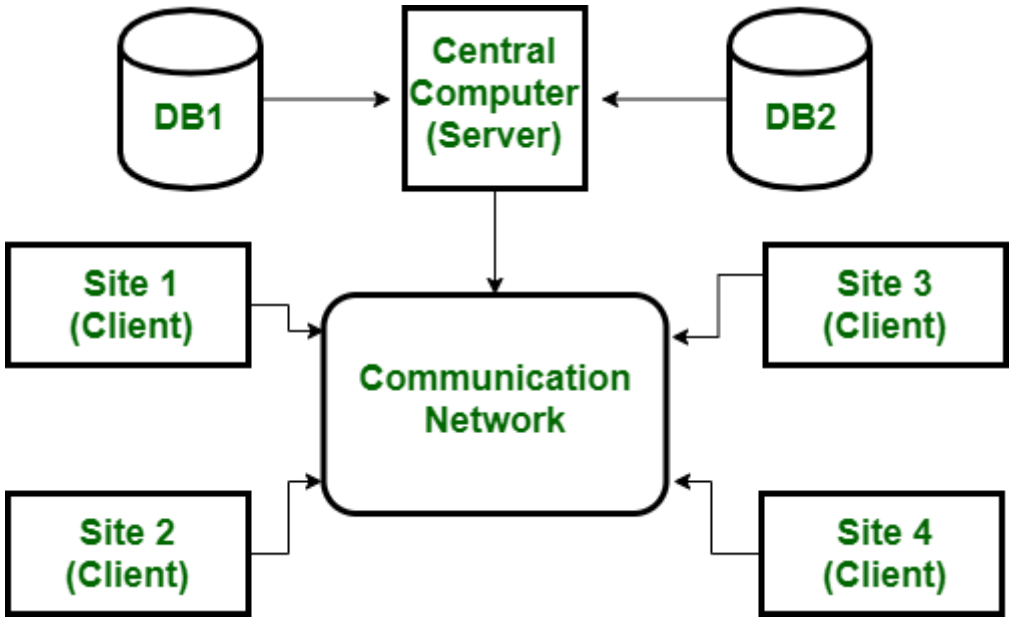

Рисунок 3.3 – Ілюстрація клієнт-серверної архітектури [14]

Оскільки технології продовжують розвиватися швидкими темпами, сучасний бізнес все більше покладається на IT, щоб успішно конкурувати в середовищі постійних змін. Таким чином, потрібна система, яка спрощує збір, обробку та використання корпоративних даних, підвищує ефективність бізнес-процесів і забезпечує конкурентоспроможність на світовому ринку.

Клієнт-серверна архітектура забезпечує високий рівень обробки, що сприяє підвищенню ефективності робочих станцій, розширенню можливостей робочих груп, дистанційному управлінню мережею та збереженню наявних інвестицій. Таким чином вона забезпечує необхідну основу для вирішення сучасних завдань інформаційного світу, що стрімко розвивається.

Клієнт-серверна архітектура має свої плюси та мінуси для сучасних цифрових споживачів. Серед переваг можна відзначити:

– централізованість системи – система зберігає всі дані та елементи керування в одному місці;

– масштабованість – архітектура має високий рівень масштабованості, організації та ефективності та дозволяє розробнику масштабувати ємність окремо клієнта та сервера;

– економічна вигода – не тільки в розробці, але й в обслуговування;

– можливість відновлення даних;

– покращена продуктивність – досягається за допомогою балансування навантаження;

– обмін ресурсами між різними платформами – клієнт-серверна архітектура не обмежується однією платформою, оскільки клієнтська частина системи може бути реалізована під кожну необхідну платформу;

– зменшення випадків реплікації даних.

Однак у клієнт-серверної архітектури є і негативні сторони:

– підвищений ризик безпеки – існує ризик занесення хробаків, вірусів або троянів на сервер через підключених клієнтів і серверів;

– вразливість сервера до атак «Відмова в обслуговуванні» (Denial of Service, DoS);

– можливість підробки чи зміни пакетів даних під час передачі;

– високі витрати на розгортання та початкове впровадження;

– відмова критично важливого сервера призведе до зупинки всієї системи;

– архітектура вразлива до фішингових атак та атак типу «Man-in-the-Middle» (MITM).

3.1.4 Архітектура, керована подіями (EDA)

Архітектура, керована подіями (EDA) – це шаблон проєктування програмного забезпечення, який дозволяє організаціям виявляти та реагувати на «події» або важливі бізнес-моменти, такі як транзакції, відвідування веб-сайтів, залишення кошика тощо в реальному або майже реальному часі (рис. 3.4). Цей підхід замінює традиційну архітектуру запит-відповідь, де служби повинні чекати відповіді, перш ніж переходити до наступного завдання. В архітектурі, керованій

подіями, потік подій визначає потік виконання, реагуючи на події та виконуючи певні дії у відповідь на них.

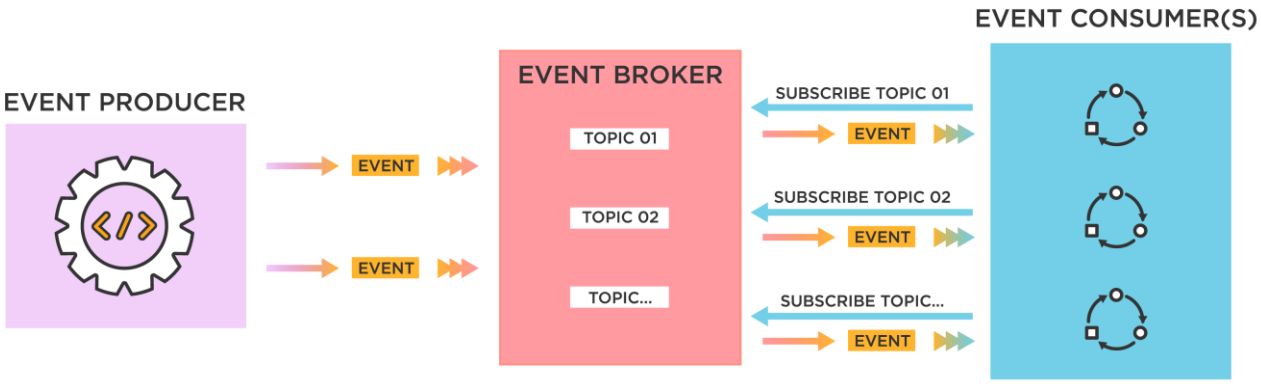

Рисунок 3.4 – Ілюстрація архітектури, керованої подіями [15]

Цю архітектуру часто називають «асинхронним» зв'язком, що означає, що відправнику та одержувачу не потрібно чекати один одного, щоб перейти до наступного завдання. Система не залежать від синхронності одного повідомлення. Наприклад, телефонний дзвінок у службу підтримки є прикладом синхронної взаємодії, подібної до традиційної архітектури запит-відповідь. Коли хтось дзвонить у службу підтримки і просить оператора щось зробити, відправник чекає відповіді особи, яка виконує завдання, і лише тоді обидві сторони завершують спілкування. У разі асинхронної взаємодії, наприклад замовлення у службі доставки, особа надсилає повідомлення і навіть не очікує відповіді, можливо, навіть не знаючи, кому саме повідомлення було надіслане в обробку, і продовжує займатися своїми справами.

До переваг архітектури, керованої подіями можна віднести:

– мала зв'язність компонентів – використання EDA призводить до слабкого зв'язку компонентів системи. Такий поділ сприяє логічному розділенню виробництва та споживання подій, дозволяючи розробникам не турбуватися про те, як події споживаються чи створюються. Мала зв'язність також дозволяє розробляти різні компоненти або мікросервіси з використанням різних мов або технологій, сприяючи масштабованості програми;

– відмовостійкість – оскільки генерація подій є асинхронною, система продовжує працювати, навіть якщо один із споживачів виходить з ладу. Події ставляться в чергу в брокері, що дозволяє споживачам забирати незавершені події після відновлення роботи;

– покращена взаємодія з користувачем у режимі реального часу – керований подіями API забезпечує покращений інтерактивний досвід для кінцевого користувача, абстрагуючи багато обов’язків, які раніше покладалися на користувачів, і задовольняючи сучасні вимоги до інтерактивності;

– ефективність – система, керована подіями, працює шляхом надсилання повідомлень до черги виробником і отримання їх споживачем, уникаючи необхідності постійного опитування для перевірки подій, що зменшує використання процесора та навантаження пропускної здатності мережі.

Недоліками ж EDA можна вважати:

– надмірна інженерія – застосування EDA може призвести до додаткових витрат через необхідність налаштувати інфраструктуру для виробників і споживачів, таку як система масового обслуговування. Також докладаються додаткові зусилля для обробки та перевірки подій замість того, щоб зосереджуватися лише на бізнес-логіці;

– непослідовна поведінка – подвійні оновлення або подвійні події можуть ускладнити їх опрацювання для системи та спричинити проблеми з часом тестування та налагодження;

– обробка помилок та усунення несправностей – наявність кількох виробників і споживачів повідомлень може вимагати встановлення та налагодження сторонніх інструментів для ефективного моніторингу потоку подій, що може бути важко зробити.

3.1.5 Вибір архітектури для інформаційної системи з підтримки обприскування за допомогою БПЛА

Метою даної роботи є розробка інформаційної системи з підтримки обприскування за допомогою БПЛА, і оскільки планується, щоб ця система надавала зручний та сучасний інструмент для аграріїв, то з цього випливає декілька факторів, які треба врахувати під час обрання архітектури.

По-перше, планується, що система буде регулярно оновлюватися, виправляючи різноманітні помилки та додаючи новий функціонал. Якщо розбивати її на багато копій, та постачати кожному клієнту окремо, тоді виникнуть труднощі з тим, щоб оновлювати усі ці відокремлені копії. З цієї точки зору підхід є нераціональним для розробника, оскільки потребує значних зусиль та перш за все значних витрат.

По-друге, для того щоб розгорнути систему на базі сільськогосподарського підприємства клієнта, необхідне додаткове обладнання (як мінімум персональний комп’ютер та локальна мережа по території підприємства), якого може не бути в наявності. Тому в цьому випадку або клієнту доведеться витрачати додаткові кошти для того, щоб придбати та встановити необхідне обладнання, або на це доведеться витрачатися розробникові. В тому чи іншому разі кінцева ціна продукту збільшиться за рахунок необхідних супутніх витрат, що аж ніяк не є вигідним жодній зі сторін.

По-третє, якщо копії системи будуть встановлюватися локально, то виникнуть значні труднощі з їх обслуговуванням. Якщо, скажімо, у клієнта під час роботи станеться якась помилка, і система неочікувано зупиниться, і якщо клієнт не зможе самостійно позбутися цієї помилки, то він звернеться у службу підтримки. В такому разі розробнику доведеться відправляти когось для того, аби виправити помилку. Але якщо підприємство клієнта знаходиться дуже далеко від офісу розробника, чи навіть в іншій країні на іншій стороні земної кулі, то клієнту доведеться дуже довго чекати на виправлення помилки (якщо він взагалі

дочекається). А за час простою пройде найважливіше та найдорожче для будь-якого фермера – час.

Останнє і не менше важливе за усе інше – якщо разом із системою планується постачати не тільки послуги з керування дронами, а й самі дрони, то наявність централізованої системи просто необхідна. Якщо ж клієнтам за додаткову плату постачати ще й самі дрони для роботи в полі за певну орендну плату та на певний період часу, то їхній моніторинг є дуже важливим. Виникає та сама ситуація, що й з самою системою – у разі виникнення помилки, чи якоїсь непередбачуваної відмови набагато зручніше буде виправити ситуацію з єдиного центру в офісі розробника, ніж їхати до клієнта особисто.

Враховуючи усі ці аспекти, а також функціональні вимоги до системи, які були висунуті в розділі 2, найбільш підходящою для даного завдання є клієнт-серверна архітектура, за рахунок наявності централізованого серверу, який буде обробляти запити від усіх клієнтів дистанційно через мережу Інтернет [16]. Окрім цього система буде використовувати вебклієнт, який буде легко доступний з будь-якого пристрою, у якого є підключення до мережі Інтернет. Це дасть змогу клієнту не залежати від спеціального обладнання і мати доступ до системи хоч з комп’ютера, хоч зі смартфона чи планшета. В той же час розробник зможе проводити оновлення системи легко і без великих витрат, і всі оновлення в той же момент будуть доступні усім клієнтам.

Також при всьому цьому буде застосовуватися концепція «тонкого клієнта». Тонким клієнтом називають такий тип клієнта, який використовує обчислювальні ресурси, розташовані не на боці самого клієнта, а на боці центрального сервера. Як правило, тонкий клієнт взаємодіє з серверним середовищем, де знаходиться уся бізнес-логіка, пам’ять та усі конфіденційні дані, необхідні користувачеві. Також є можливість підключати тонкі клієнти до серверів, розташованих у хмарному середовищі.

У багатьох випадках тонкий клієнт стає ефективною альтернативою персональному комп'ютеру (ПК). Це може бути особливо корисним рішенням, оскільки дозволяє розробнику надавати користувачам робочий інтерфейс без

прив’язки до їхнього обладнання. Крім того, це відкриває можливість централізованого управління рішеннями безпеки, захищаючи сервер, до якого підключаються різноманітні тонкі клієнти. Таким чином викрасти будь-які дані стає майже неможливо, оскільки вони не зберігаються на самому клієнтів, а на центральному сервері з відповідно обладнаною системою безпеки.

Отже, після обрання архітектури застосунку наступним кроком буде її деталізація на усіх рівнях. А рівнів цих буде три: рівень клієнта, рівень сервера та рівень бази даних. Подібний підхід називається трирівневою архітектурою (Three-Tier Architecture) і є популярним розширенням клієнт-серверної архітектури. Згідно з визначенням трирівневої архітектури три рівні також мають назви: рівень презентації (presentation tier), рівень застосунку (application tier) та рівень даних (data tier).

Перший рівень – рівень презентації – це інтерфейс, за допомогою якого користувач взаємодіє з системою. Цей рівень збирає інформацію від користувача, передає її на наступний рівень (рівень застосунку) та відображає відповіді, які надійшли з нього. Найбільш поширеними представниками цього рівня є сторінки веб-браузера. В такому разі рівень презентації розробляється з використанням HTML, CSS і JavaScript.

Другий рівень – рівень застосунку – є найголовнішим, бо на ньому зберігається вся логіка системи. Саме тому цей рівень ще відомий як логічний рівень. Після надходження на цей рівень даних з попереднього рівня, вони обробляються бізнес-логікою, після чого формуються відповіді, які відправляються назад на рівень презентації. Також цей рівень здійснює взаємодію з рівнем даних, додаючи, зчитуючи, змінюючи та видаляючи дані до та з БД.

Останній за списком, але не останній за значенням, третій рівень – рівень даних – є тим місцем, в якому зберігається уся інформація системи, усі важливі дані, які необхідні для її роботи. Так, наприклад, тут можуть зберігатися дані про користувачів системи, історія їхніх дій, дані про товари чи обладнання, якщо таке в системі використовується, тощо. Найчастіше цей рівень представлений за

допомогою БД, і зазвичай БД є реляційною, хоча останнім часом все більш популярними стають нереляційні аналоги.

Реляційні БД використовують для зберігання даних структуру, створену на основі таблиць. Вони організовують дані в таблиці з чітко визначеними зв’язками між окремими елементами даних (сутностями), що полегшує навігацію між цими зв’язками. Шляхом упорядкування таблиць, вимог до полів та зв’язків між ними певним чином створюється так звана схема БД.

Мова програмування, яка використовується для створення та зв’язку з реляційними БД, називається мовою структурованих запитів (structured query language, або SQL), і часто використовується для позначення реляційних БД. SQL дозволяє отримувати дані, виконувати запити та маніпулювати даними, включаючи оновлення, видалення та додавання нових записів.

Раннє впровадження та широке використання SQL БД є причиною їх постійної привабливості як технології керування даними. А також немаловажливим фактором популярності цього типу БД є їхня більша зрозумілість, особливо для новачків, які тільки починають знайомитися з концепцією БД. Тому SQL БД, як правило, стають першим типом БД, який вивчають нові спеціалісти.

Нереляційні БД, які також часто називають NoSQL, відрізняється від реляційних тим, що вони не дотримуються табличної схеми рядок-стовпець. Натомість механізм зберігання, який вони використовують, адаптований до унікальних характеристик даних, які вони контролюють.

На відміну від реляційних БД, які використовують лише SQL, NoSQL (як випливає з назви, «Not only SQL» – «не лише SQL») підтримують інші мови запитів.

Існує чотири основні та загальноприйняті категорії NoSQL БД:

– БД, орієнтовані на документи – БД, які іноді називають сховищами документів, призначені для зберігання, отримання та обробки даних, орієнтованих на документи. Як правило, кожен ключ пов’язаний з документом – складною структурою даних;

– сховища ключ-значення – у цих БД використовуються різні ключі, кожен з яких пов’язаний із певним значенням у колекції – порівняльною концепцією може бути, наприклад, словникова БД. Цей вид NoSQL БД є одним з найпростіших;

– широкостовпчикове сховище (з англ. wide-column store) – БД, які на подобу до реляційних, використовують таблиці, рядки та стовпці, проте в одній таблиці назви та формати стовпців можуть відрізнятися від рядка до рядка;

– графові сховища – БД, які зберігають дані у форматі вузлів, ребер та атрибутів, формуючи при цьому зв’язний граф, який може бути використаний для семантичних запитів.

Рішення вибору того чи іншого виду БД лишається за розробником. Реляційні БД підходять у випадку наявності чіткої структури даних та явних і зрозумілих залежностей між ними. В той час як нереляційні БД можуть надати додаткової гнучкості у випадку, коли структура даних, які будуть використовуватися в системі, ще не відома на момент розробки, чи коли завчасно відомо, що дані точно будуть різного формату. NoSQL БД можуть бути помічними, коли необхідно досягти більшої швидкодії системи, особливо, коли мова йде про постійну обробку великого масиву даних. В такому випадку БД можна завчасно спроєктувати під запити, які до неї будуть надсилатися.

Отже, оскільки для системи, що розробляється, дуже важливим фактором є її централізованість, то вибір трирівневої архітектури є найбільш вдалим рішенням. Окрім того ця архітектура не тільки задовольняє вимогам, а і є доволі зрозумілою та простою в плані розробки.

3.2 Опис функціональної складової системи

Для того аби визначитися з конкретними технологіями, які будуть використовуватися системою, та на яких вона буде побудована, необхідно спочатку зрозуміти, які функції ця система буде виконувати. Для цього необхідно визначити, по-перше, хто буде використовувати систему, хто буде користувачами

та які ролі будуть вони виконувати. А по-друге, яким самим чином користувачі будуть взаємодіяти із системою, який функціонал будуть використовувати, та в якій послідовності.

А для того аби визначити усі ці аспекти, дуже помічним буде використати UML діаграми. UML (з англ. unified modeling language) або ж уніфікована мова моделювання – це спеціально побудована мова, яка створена специфічно для моделювання програмних рішень, структур застосунків чи систем, їхньої поведінки та бізнес-процесів. По суті UML не є мовою у звичному розумінні цього слова, а натомість це сукупність нотацій, які використовуються для побудови діаграм та схем з використанням блоків, які описують залежності в системі, її поведінку та процеси. UML діаграми діляться на два основні види: структурні та поведінкові (рис. 3.5).

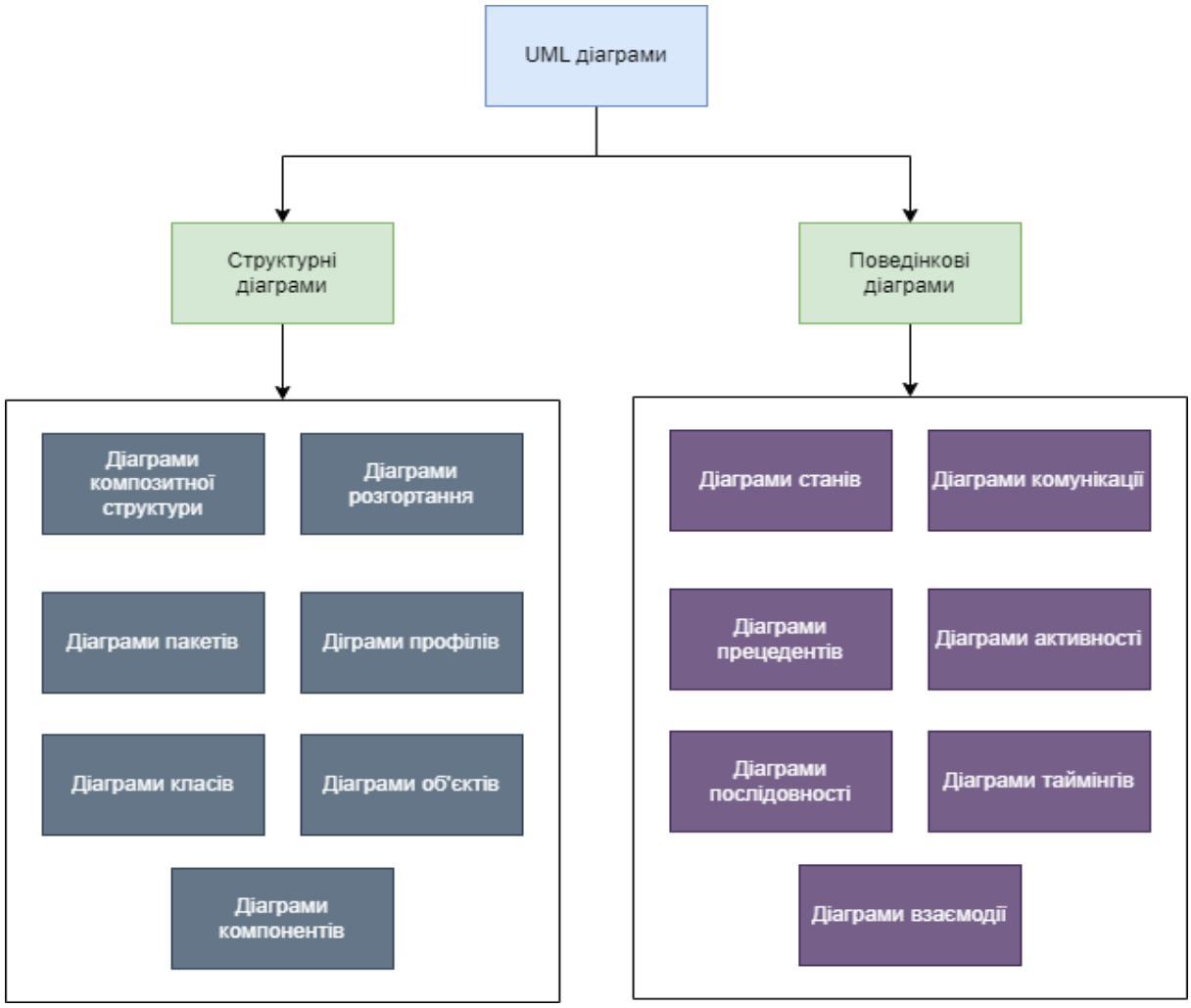

Рисунок 3.5 – Різновиди UML діаграм

Кожен з видів відіграє свою важливу функцію. В той час як структурні UML діаграми описують структуру самої системи, основні її компоненти та зв’язки між ними, структуру даних в системі, то поведінкові діаграми описують, відповідно, поведінку цієї системи, які процеси в ній відбувається, в якому порядку вони відбуваються та у яких часових проміжках, а також пояснюють, як саме користувачі можуть взаємодіяти із системою, та як система буде їм відповідати.

Для початку використання UML найперше, з чим потрібно визначитися, це які в системі будуть актори – дійові особи, або інші системи чи підсистеми, які взаємодіють з основною системою. Найголовнішим актором, який і буде найбільше взаємодіяти із системою, буде користувач-клієнт. Також із системою будуть взаємодіяти адміністратор та банківська система.

Наступним кроком буде визначення переліку прецедентів – дій, які актори можуть проводити, взаємодіючи із системою. Цей перелік представлено в таблиці 3.1.

Таблиця 3.1 – Перелік акторів та їх прецедентів

Актор	Прецеденти
Клієнт	- створює обліковий запис; - авторизується в системі; - вибір плану підписки на сервіс; - оплата підписки на сервіс; - зчитує наявні в системі об’єкти на карті; - зберігає в системі новостворені об’єкти на карті; - видаляє з системи об’єкти на карті; - наносить на карту нові об’єкти; - додає моделі дронів; - редагує моделі дронів; - видаляє моделі дронів; - додає дрони; - редагує дрони; - видаляє дрони; - створює РП; - редагує РП; - видаляє РП;

Актор	Прецеденти
	- запускає виконання РП.
Адміністратор	- додає плани підписки на сервіс; - редагує плани підписки на сервіс; - видаляє плани підписки на сервіс; - видаляє користувачів; - переглядає звіти по роботі системи.
Банківська система	- проводить оплату підписки на сервіс.

В такому разі, якщо розписати більш розширено, то користувач-клієнт повинен мати можливість створювати новий обліковий запис і авторизуватись в системі. Для цього необхідно передбачити форму для реєстрації та авторизації, а також відповідний функціонал, який буде передавати дані з форми на сервер, а той у свою чергу шифруватиме пароль, та зберігатиме внесені користувачем дані разом із зашифрованим паролем до БД. Також для клієнта має бути доступна можливість вибору одного із запропонованих планів підписки на сервіс, оскільки комерційна складова – одна з найбільш пріоритетних задач цієї системи.

Після отримання доступу до сервісу користувач-клієнт зможе взаємодіяти з картою, додаючи та видаляючи до (з) нею об’єкти, зберігати додані об’єкти до БД, щоб потім мати можливість їх автоматичного підвантаження під час роботи. Також клієнту має бути доступна можливість додавання, редагування та видалення дронів та їхніх моделей. Це має надати клієнту свободу у виборі дронів, а також можливість надавати кожному дрону унікального імені, навіть якщо вони однакової моделі.

І на останок клієнт повинен мати можливість створювати, редагувати та видаляти робочі плани (РП). Відповідно для цього він у відповідній формі вибирає робочу область (РО), точку завантаження (ТЗ) та дрона і вводить необхідні параметри для того, щоб створити РП. Або він може вибрати вже існуючий РП зі списку, щоб відредагувати чи видалити його.

Функцій же користувача-адміністратора не має бути так багато. Його основними обов’язками мають бути забезпечення сервісу оновленими планами підписки, перегляд звітності роботи сервісу та видалення облікових записів, якщо

на це є підстави (порушення умов користування, відсутність активності понад один календарний рік, підозріла активність тощо).

Актор «банківська система» тут – це зовнішня система банку, до якої буде звертатися система, яка розробляється в цій роботі, для проведення оплати за підписку на сервіс.

Усі зазначені прецеденти та усіх акторів можна переглянути на діаграмі прецедентів (додаток А), де відображено повну картину взаємодії акторів із системою.

Але узагальнена діаграма прецедентів покриває лише основних функціонал, з яким будуть взаємодіяти користувачі та сторонні системи. Так, наприклад, користувач-клієнт проходячи реєстрацію в системі може отримати помилку, у зв’язку з наявністю користувача із зазначеним у формі логіном. Таким же чином помилка може виникнути під час авторизації, в разі невідповідності логіна чи пароля даним, які зберігаються в системі. Додатково під час авторизації система проводить порівняння хешу введеного пароля з тим, який закріплений за користувачем в системі.

Також під час вибору клієнтом плану підписки система проводить для нього перевірку наявності нових планів, або оновлень вже наявних. Окрім того, під час роботи з дронами, їх моделями та РП можуть статися різні помилки, які потрібно владнати та повідомити про них користувача. Найчастіше помилки можуть виникати під час створення, редагування та видалення певних об’єктів. Також під час додавання об’єктів система перевіряє правильність вводу усіх полів та параметрів, щоб вони відповідали встановленим шаблонам. А під час створення РП система також обраховує оптимальну ТОП, під час розрахунку якої також можуть виникнути помилки.

Окрім цього під час нанесення на карту нових об’єктів можна обирати, який саме об’єкт буде доданий (РО чи ТЗ), а також є можливість відміни останньої дії, видалення останнього доданого на карту об’єкта та видалення усіх новододаних на карту об’єктів. Під час же збереження об’єктів з карти до системи необхідно надати унікальні назви чим об’єктам. Під час цього процесу якраз і перевіряється

умова унікальності. Також під час роботи з картою та її об’єктами їх можна за потреби виділити та видалити. Окрім усього цього система через певний інтервал часу проводить автоматичне оновлення усіх об’єктів на карті. Для кожного активного користувача вона порівнює вже існуючі на карті об’єкти з тими, які знаходяться в системі, та додає на карту ті, яких там ще не має.

Користувач-клієнт, як вже зазначалося раніше, може проводити оплату за обраним планом підписки. Під час цього відбувається взаємодія із сторонньою системою банка, яка може заблокувати платіж з тих чи інших причин (наприклад, за відсутності необхідної суми на рахунку клієнта). В свою чергу в системі, що розробляється в цій роботі, може відбутися своя внутрішня помилка обробки запиту на оплату, що також буде відображено користувачеві.

Для більш детального огляду усіх можливих взаємодій з урахуванням різного роду помилок, підтверджень, перевірок та розширеної логіки взаємодії дивись розширену діаграму прецедентів у додатку Б.

3.3 Послідовність взаємодії користувачів з системою

Розуміння та моделювання порядку, в якому користувачі взаємодіють із системою, є ключовим компонентом проєктування інформаційної системи. Створюючи систему для підтримки використання БПЛА для обприскування сільськогосподарських полів, важливо врахувати вимоги користувачів і створити інтуїтивно зрозумілий та ефективний інтерфейс користувача. А для того, щоб цього досягти не вийде обмежитися лише діаграмою прецедентів, оскільки вона визначає лише перелік можливих взаємодій, але не пояснює самого процесу взаємодії. Тому для з’ясування усіх можливих варіантів взаємодії першим ділом необхідно змоделювати поведінку користувача, який починає користування системою.

Очікується, що користувач-клієнт взаємодіятиме із системою різними способами та з різними завданнями. Глобально ця взаємодія стосується обприскування полів та передбачає створення та керування робочими планами

(РП), конфігурацію налаштувань дронів і моніторинг обробки полів у реальному часі. Але якщо заглибитися детальніше, то вийде побачити, що для досягнення необхідного результату користувачу знадобиться пройти більше кроків.

Для візуалізації порядку взаємодії користувача із системою можна скористатися діаграмою активності. З її допомогою можна прояснити логіку взаємодії та визначити можливі точки оптимізації, визначивши ключові фази взаємодії та порядок дій.

Діаграма активності – це більш складна форма блок-схеми, яка ілюструє, як діяльність перетікає від однієї активності до іншої. Діаграми активності показують, як різні рівні абстракції можна використовувати для зображення того, як дії координуються для створення послуги. Як правило, операція повинна досягати кількох цілей, які потрібно координувати, або вона повинна визначити, як події в одному варіанті використання співвідносяться одна з одною, особливо у тих випадках використання, коли може бути збіг у діях, які потрібно координувати. Їх також можна використовувати для моделювання координації між прецедентами з діаграми прецедентів для зображення бізнес-процесів.

Діаграму активності для системи, що розробляється, побудовану на основі діаграми прецедентів та загальних функціональних вимогах, можна побачити у додатку В.

Відповідно до діаграми процес роботи для користувача може виглядати наступним чином. Клієнт після входу в систему стикається з формою авторизації. Якщо у нього ще немає зареєстрованого облікового запису, він може пройти на сторінку реєстрації, де заповнивши форму, зможе створити новий обліковий запис. Після успішного створення облікового запису система поверне клієнта на сторінку авторизації, де він зможе ввести логін та пароль і продовжити роботу.

Увійшовши до свого акаунта, клієнт зможе побачити сторінку роботи з картою, і зможе з нею взаємодіяти з інструментами нанесення об’єктів на карту та їх збереження/видалення, але йому не буде доступна вкладка роботи з РП, а також сторінка з роботою з дронами. А для того щоб розблокувати цей функціонал, клієнту потрібно обрати план підписки на сервіс та здійснити його оплату. Після

успішної оплати функціонал стане доступним і перед клієнтом відкриється широкий вибір.

Клієнт зможе вибрати одну з функцій, зображених на діаграмі у додатку А, здійснити порядок дій для обраної функції відповідно до діаграми з додатку В, після чого процес роботи не припиняється, бо він є зацикленим, і клієнт може провести скільки він забажає часу в роботі, перед тим як вийти з облікового запису.

3.4 Засоби розробки системи

Оскільки одними з найголовніших вимог до системи, що розробляється, було зазначено можливість доступу до сервісу з будь-якого пристрою (ПК, смартфону чи планшету) та централізованість системи, а за архітектуру було взято клієнт-серверний підхід, то найбільш вдалим варіантом буде розробляти систему, доступ до якої буде здійснюватися через вебпортал.

Наразі існує багато різних платформ для розробки вебзастосунків. Серед найбільш популярних є Ruby, AngularJS, Node.js, ASP.NET, React.js, Django, а також є й такі, що набирають популярність – Express.js, Flask, Vue.js, Laravel, Symfony та інші. Але для розробки системи в цій роботі обрано платформу ASP.NET Core.

ASP.NET Core – це фреймворк від компанії Microsoft, який є логічним нащадком фреймворку ASP.NET. Головними його особливостями зазначаються кросплатформенність, висока продуктивність та відкритий вихідний код [17]. Серед переваг цієї платформи можна відзначити:

– кросплатформенність – платформа є сумісною з більшістю операційних систем як, наприклад, Windows, Linux чи macOS;

– простота в обслуговуванні застосунків – через те, що мова програмування платформи C# є строго типізованою, це дозволяє виявляти більшість дефектів ще на ранній стадії розробки, а такі інструменти як Visual Studio Intellisense допомагають підтримувати кодову базу проєкту;

– підтримка Веб-API – Веб-API є складовою частиною фреймворку і його можна легко активувати в системі, що розробляється;

– легкість масштабування та докеризації – застосунки, написані за допомогою цього фреймворку, можна доволі легко масштабувати, а також їх можна запускати через Docker, що полегшує контейнеризацію та створення інфраструктури для архітектури мікросервісів;

– відкритий код – це означає, що будь-хто може внести зміни до вихідного коду, які пришвидшать розробку чи покращать загальну якість продукту, і якщо ці зміни будуть дієві та сприйняті загалом, то вони можуть бути додані до наступної версії фреймворку офіційно;

– продуктивність – фреймворк забезпечує непогані функції асинхронності програмування, що значно підвищує продуктивність у порівнянні з «не Core» версією (рис. 3.6) [18].

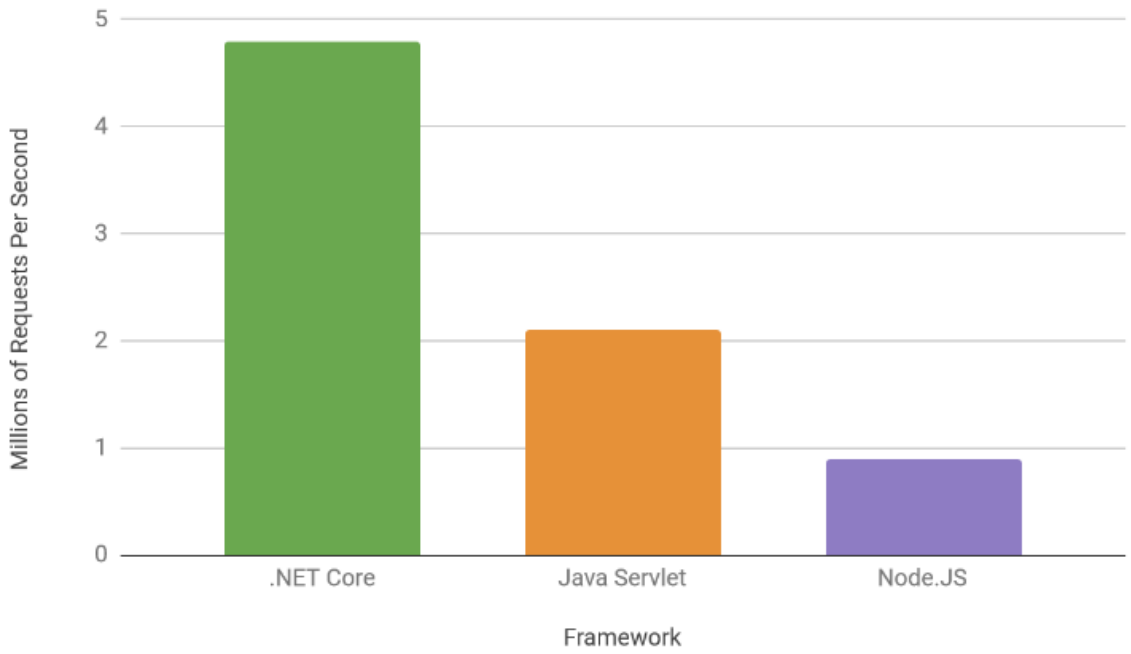

Рисунок 3.6 – Порівняння продуктивності .NET Core з іншими фреймворками [18]

Також для розробки системи буде використовуватися шаблон проєктування MVC (model-view-controller). MVC є однією з найпопулярніших галузевих стандартних структур веброзробки для створення масштабованих і адаптованих

проєктів. За допомогою MVC систему можна розділити на три основні логічні компоненти: модель (model), представлення (view) та контролер (controller). Усі ці частини створені для роботи з певними аспектами розробки системи.

Усю логіку даних, що взаємодіють із користувачем, обробляє компонент моделі. Також цей компонент може включати будь-які додаткові дані, що стосуються бізнес-логіки, а також дані, якими обмінюються компоненти представлення і контролера. Представлення в свою чергу реалізує всю логіку інтерфейсу користувача системи. Контролери обробляють усі вхідні запити та бізнес-логіку, служачи інтерфейсом між компонентами моделлю та представленням. Вони працюють із представленнями, щоб відобразити готовий продукт після зміни даних за допомогою компонента моделі.

Фреймворк ASP.NET Core також включає можливість реалізації шаблону MVC. Серед переваг ASP.NET Core MVC можна виділити наступне:

– ідеально підходить для створення складних, але не надто важких програм;

– пропонує структуру, яку легко замінити та конфігурувати;

– оскільки макетні об’єкти можна використовувати для тестування компонентів, а інтерфейси можна використовувати для проєктування компонентів, структура MVC покращує як розробку, керовану тестуванням, так і можливість тестування програми;

– підтримує всі функції, які зараз доступні в ASP.NET, включаючи підписки, елементи керування користувачами, головні сторінки, зв’язування даних, авторизацію/автентифікацію та маршрутизацію;

– усуває потребу в концепції стану перегляду (view state), який включений в ASP.NET, для створення легких програм із повним контролем розробника [19].

Окрім фреймворку ASP.NET Core для розробки певних компонентів буде розумно залучити мову програмування Python. Python – це мова, яка доволі проста в розумінні та лаконічна в написанні. Найсильніша перевага цієї мови – її відкритість коду та наявність користувацьких бібліотек. Це призводить до того, що цю мову можна використовувати майже для будь-яких цілей. Також завдяки розробленим спільнотою бібліотекам за допомогою Python можна проводити

складні математичні обчислення, а також будувати комплексні математичні та фізичні моделі. Окрім того, усі дані можна за необхідності конвертувати в необхідні формати, а також залучати різноманітні бібліотеки для їх графічного відображення. Це дає вагому перевагу не тільки для покращення користувацького досвіду, а й полегшує виявлення помилок для розробників. Так наприклад, під час програмування алгоритму пошуку найкоротшого шляху розробник може стикнутися із ситуацією, коли програма запускається і відпрацьовує без проблем, але видає некоректний результат. В такому разі на допомогу приходить графічна візуалізація, яка може показати, в якому саме місці алгоритм дав збій. Отже, саме через перелічені характеристики мова Python з її користувацькими бібліотеками якнайкраще підійде для розробки модуля побудови ТОП.

Також немаловажливим є надання користувачу-клієнту можливості працювати з картою, на якій буде використовуватися шар супутникових знімків. Оскільки основне призначення карти – надавати користувачу можливість створення РО та ТЗ у зручному форматі малювання, замість незручного та неоптимального методу вводу цих об’єктів за допомогою самих лише координат, життєво важливо мати на картах саме знімки місцевості, по яким вже можна буде будувати об’єкти.

Тому до частини представлення системи було обрано включити інструмент відображення карт, який буде забезпечуватися за допомогою OpenLayers API. OpenLayers – це набір інструментів JavaScript із відкритим кодом, який відображає динамічні карти з векторних даних і фрагментів карт. За допомогою OpenLayers можна легко інтегруйте динамічні карти на будь-яку вебсторінку. Бібліотека може відображати фрагменти карти, векторні дані та маркери, завантажені з будь-якого джерела. Метою розробки OpenLayers є розширення застосування всіх типів географічних даних. API опубліковано за ліцензією BSD із 2 пунктів (часто називають FreeBSD) і є абсолютно безкоштовним [20].

Оскільки OpenLayers використовує платформу Node.js для роботи, важливо визначитися з тим, яким чином можна інтегрувати необхідний функціонал до

системи, що розробляється на ASP.NET Core. Але спершу треба зрозуміти, що собою являє Node.js.

Node.js – це кросплатформне середовище виконання JavaScript із відкритим кодом, яке є популярним інструментом як для вибудови клієнтської вебчастини системи, так і для проєктування серверної. Node.js працює на рушії V8 JavaScript, який є основою Google Chrome, за межами самого браузера. Це забезпечує високу його продуктивність.

Середовище Node.js не запускає новий потік для кожного запиту; замість цього він працює в єдиному процесі. Стандартна бібліотека Node.js, яка включає набір асинхронних примітивів введення-виведення, уникає блокування в коді JavaScript. Крім того, бібліотеки, створені в Node.js, часто розроблені для уникнення блокування, тому блокування є скоріше винятком, ніж правилом.

Обробляючи запити, Node.js продовжує свої операції вводу-виводу, такі як читання з файлової системи, бази даних або мережі, замість того, щоб блокувати потік та марнувати цикли ЦП, чекаючи. Це усуває необхідність керувати паралельністю потоків, яка може бути основною причиною помилок, і дозволяє Node.js обробляти тисячі одночасних з’єднань з одним сервером [21].

І тут постає проблема інтеграції OpenLayers до системи, що розробляється. Оскільки робота Node.js потребує запуску окремого середовища, яке є абсолютно самостійним та буде конфліктувати з основним середовищем ASP.NET Core, потрібно знайти спосіб, як можна було б використовувати модулі Node.js поза їх основним середовищем.

В такому випадку на допомогу можуть прийти програми-пакувальники. Головна їхня задача – сканування наданих їм файлів на предмет наявних залежностей, пошук усіх залежних файлів та формування єдиного файлу-збірки з усім функціоналом, який був розділений по модулях, на які були посилання. Прикладом подібного пакувальника може бути Webpack.

Webpack – це, по суті, пакувальник статичних модулів для сучасних програм JavaScript. Після обробки програми Webpack створює внутрішню структуру залежностей на основі однієї або кількох точок входу. Потім він

об’єднує всі модулі, необхідні для вашого проєкту, в один або кілька пакетів, які є статичними активами, які використовуються для обслуговування вашого вмісту. Цей процес умовно проілюстрований на рисунку 3.7.

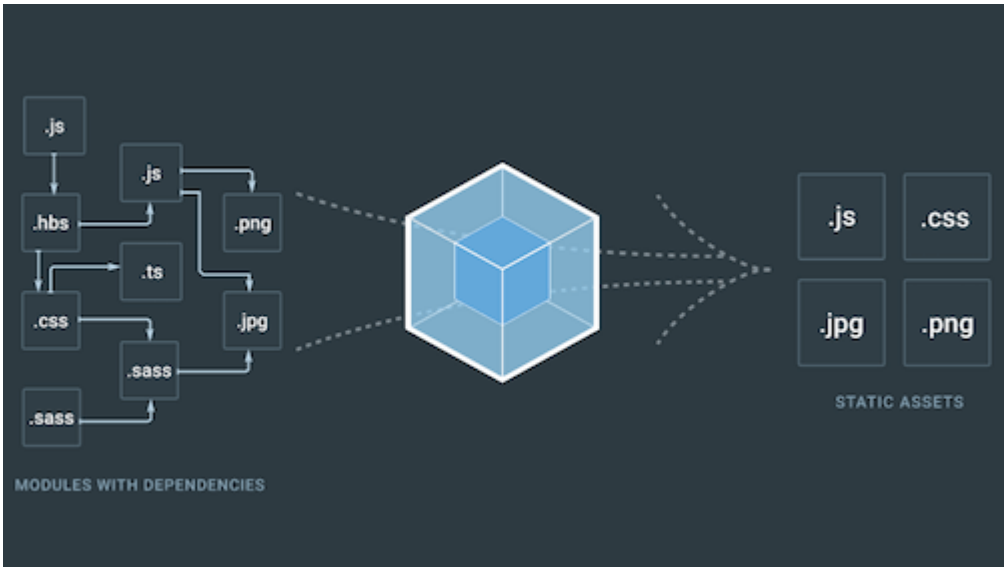

Рисунок 3.7 – Принцип роботи пакувальника модулів Webpack [22]

І останнім за списком, але не за значенням, буде вибір бази даних, яка буде використовуватися в системі, що розробляється. Як вже зазначалося у пункті 3.1, існує два глобальні типи БД – реляційні (або ж SQL) та нереляційні (NoSQL). Кожен з типів має свої переваги та недоліки, і один є більш пристосованим до певних задач ніж інший, і навпаки.

Так у дослідженні [23] від 2023 року було проведено метааналіз статей, в яких розглядалася різниця між SQL та NoSQL БД. За результатами цього аналізу було визначено, що ні один, ні другий типи БД не мають явних переваг. Натомість кожний тип може бути більш корисним та більш продуктивним в одних випадках, але менш продуктивним та менш корисним в інших. Так, наприклад, автори дослідження зазначають, що в разі наявності чіткої стандартизації та узгодженості даних більш вдалим рішенням буде обрати SQL БД. В той же час за наявності протилежної ситуації, коли дані є неструктурованими, а їхня доступність є найбільшою вимогою, то варто спробувати використати NoSQL БД. Також автори цього дослідження зазначають, що NoSQL добре підходить для роботи з

великими обсягами даних, у випадках коли йдеться про аналітику у галузі великих даних (big data). Також вказано, що при залучення геопросторових даних, реляційні БД мають кращі показники масштабованості, ніж нереляційні. Втім коли мова йде про величезні обсяги геопросторових даних, то NoSQL БД показують швидший час відповіді на запити.

В іншому дослідження [24], проведеному в 2018 році, автори приходять до схожих висновків: за потреби дотримуватися властивостей ACID (atomicity, consistency, isolation, and durability – атомарність, послідовність, ізоляція та довговічність) краще вибирати реляційні БД, а для гнучкості моделі та зберігання великого обсягу даних у різних форматах краще підійдуть NoSQL БД.

Втім автори зазначають, що NoSQL можуть бути більш вразливими в плані безпеки даних, і тому для БД цього типу часто залучають додаткові зовнішні методи безпеки. Але також в цій роботі зазначається, що робота в цьому напрямку активно ведеться.

Враховуючи зазначену інформацію можна зробити висновок, що для системи, яка розробляється в цій роботі, найкраще підійде БД саме NoSQL типу, оскільки ці БД мають найкращі показники масштабованості, коли мова заходить за додавання нового функціоналу до вже розгорнутого проєкту, а також тому що система буде оперувати великою кількістю геопросторових даних.

І якщо вже з типом БД все визначено, тоді тепер слід розглянути, яка саме NoSQL БД краще за всі підійде для використання в системі, що розробляється. Як вже зазначалося в 3.1.5, наразі існує чотири види NoSQL БД: БД ключ-значення, широкостовпчикові, документо-орієнтовані та графові БД. А дати відповідь на питання, який же вид найкраще підійде допоможуть сторонні дослідження.

Так, наприклад, у дослідженні [25] виділяються наступні особливості кожного виду:

– БД ключ-значення – уникають таких функцій, як спеціальні запити, об'єднання та агрегатні операції на користь високої масштабованості, а не узгодженості. Їхні сильні сторони – вибір масового зберігання, швидкий пошук і

висока паралельність. Тим не менш, відсутність схеми ускладнює процес розробки унікальних представлень даних;

– широкостовпчикові БД – пропонують високу масштабованість зберігання даних, організовуючи дані для швидкого агрегування з меншою активністю введення-виведення. Порядок сортування сімейства стовпців визначає порядок зберігання, що сприяє ефективній обробці та пошуку даних;

– документно-орієнтовані БД – коли модель предметної області можна розділити та розділити на кілька документів, зберігання документів працює ефективно. Якщо БД буде містити багато зв’язків і стандартизації, слід уникати використання цього виду БД;

– графові БД – кожен вузол у графових БД має прямий покажчик на наступний вузол завдяки суміжності без індексів. Завдяки швидкому переходу між мільйонами записів цей метод підкреслює зв’язки між даними. Графічні БД працюють швидше, ніж реляційні бази даних, оскільки вони ефективно зберігають напівструктуровані дані без необхідності використання схем і описують запити як обхід. Також вони пропонують функцію відкату, зручні для візуального зображення на дошці, сумісні з ACID і легко розширюються.

Інше дослідження [26] проводить порівняння вже неопосередковано між декількома NoSQL БД. У цьому дослідженні проводиться аналіз ефективності роботи Cassandra та HBase, а також їхнє порівняння з реляційною MySQL для отримання контрольних значень. У результаті аналізу було отримано висновок, що в той час, як приблизно при 7000 операціях читання або запису на секунду MySQL перестає відповідати (час затримки стає надто великим), Hbase та Cassandra справляються і зі значно більшими навантаженнями. А Hbase ще й має доволі стабільні показники затримки, незважаючи на навантаження, що пояснюється тим, що він проводить запис у пам’ять (а не безпосередньо на диск, як інші продукти).

Також у дослідженні [27] було проведено порівняльний аналіз 5 нереляційних БД: Redis, MongoDB, Couchbase, Cassandra та Hbase. Вони порівнювали у трьох сценаріях: операції зчитування/оновлення у співвідношенні

50/50, тільки операції зчитування та тільки операції оновлення. Таким чином в експерименті з фіксованим числом записів розглянуті БД показали різну швидкість. За цим показником їх можна розмістити у порядку зростання часу обробки таким чином: Redis, MongoDB, Cassandra, Hbase, Couchbase. У разі використання різних сценаріїв список зазнає незначної зміни: Redis, MongoDB, Couchbase, Hbase, Cassandra. В іншому експерименті було проведено аналіз із залученням різного числа записів та замірянням кількості операцій на секунду. В такому випадку БД можна розташувати у порядку зменшення кількості операцій на секунду наступним чином: Redis, Couchbase, Cassandra, MongoDB, Hbase. Але у разі використання сценарію з розподілом операцій 50/50 було отримано трохи інакший результат: Redis, Couchbase, Cassandra, Hbase, MongoDB.

Але з усього широкого переліку нереляційних БД для використання у системі, що розробляється в цій роботі, було обрано DynamoDB через ряд факторів, які яскраво узагальнює та підсумовує дослідження [28]. В цьому дослідженні зазначається, що DynamoDB має непогану масштабованість і дозволяє власникам послуг масштабувати вгору та вниз залежно від поточного навантаження запитів. DynamoDB також дозволяє власникам послуг налаштувати свою систему зберігання відповідно до бажаної продуктивності, довговічності та узгодженості SLA.

Також в цій роботі даються поради, щодо того, в яких випадках непоганим варіантом буде подумати над використанням саме DynamoDB під час розробки нової системи. Так якщо ви не збираєтеся виділяти навіть одного співробітника для керування серверами БД, чи якщо у вас відсутній бюджет для обслуговування виділених серверів БД, чи якщо ви збираєтеся інтегруватися з іншими сервісами Amazon, то вибір DynamoDB буде підходящим. І саме через те, що система, яка розробляється в цій роботі, буде невеликим стартап-проєктом з дуже малою командою і таким самим фінансуванням, вибір DynamoDB є досить обґрунтованим.

3.5 Вибір бази даних

Компанії широко використовують дані, щоб керувати своєю діяльністю та робити мудрі висновки. Ефективна архітектура БД має важливе значення для забезпечення безпечного, точного та ефективного керування даними за лаштунками.

Ефективний дизайн бази даних допомагає підвищити загальну ефективність шляхом спрощення процедур керування даними. Існує декілька важливих аспекти, для яких гарна архітектура БД є важливою.

Перше – збереження цілісності та точності даних. Компанії потребують точних і надійних даних для прийняття рішень. Такі методи, як перевірка даних і обмеження, використовуються в добре розроблених БД, щоб забезпечити дотримання правил і гарантувати точність записаних даних. Компанії можуть покладатися на свої БД для отримання точної інформації, що покращує процес прийняття рішень, уникаючи неточностей і суперечностей у даних.

Друге – підвищення масштабованості та продуктивності. Ефективний дизайн БД також значно впливає на масштабованість і продуктивність системи. Ефективним способом отримання та обробки даних є добре розроблені БД з використанням таких методів, як індексування, оптимізація запитів і стандартизація даних. Це покращує взаємодію з користувачем, прискорює час реакції та здатність керувати зростаючими обсягами даних без шкоди для продуктивності.

Третє – захист приватної інформації. Сильні функції безпеки є частиною ефективної архітектури БД, яка захищає особисту інформацію від небажаного доступу. Дані повинні бути захищені як під час передачі, так і в стані спокою завдяки використанню алгоритмів шифрування, контролю доступу та методів автентифікації. Підприємства можуть зменшити ризик витоку даних і захистити свій бренд, включивши ці заходи безпеки в дизайн БД.

Четверте – виконання нормативних зобов'язань. Архітектура БД має важливе значення для дотримання нормативних стандартів. Процедури захисту,

зберігання та перевірки даних можуть вимагатися навіть на законодавчому рівні в тій чи іншій країні. Правильний дизайн БД гарантує, що компанії зможуть дотримуватися цих правил і уникнути проблем із законом і грошима.

П’яте – надання підтримки бізнес-аналітики. Основу для ефективного пошуку, аналізу та звітності даних забезпечує саме дизайн БД. Використовуючи технології бізнес-аналітики та аналітики, добре організовані БД з відповідним моделюванням даних і дизайном схем дозволяють компаніям отримувати глибоку інформацію. Завдяки стратегічному дизайну БД підприємства можуть повністю використовувати свої дані для прийняття обґрунтованих стратегічних рішень.

Шосте – заохочення процесу прийняття рішень. Особам, які приймають рішення, потрібна точна та своєчасна інформація. За допомогою таких методів, як сховище даних та інтеграція, належна архітектура БД робить відповідні дані більш доступними. Крім того, особи, які приймають рішення, можуть покладатися на єдине джерело правдивої інформації завдяки узгодженості даних, створеній ефективним дизайном БД, що сприяє впевненому та обізнаному прийняттю рішень.

Отже, є очевидним той факт, що хороший дизайн БД є не тільки запорукою швидкості та масштабованості системи, але й підґрунтям для значно ширшого переліку речей.

Існує два підходи до проєктування структури БД: спочатку БД (Database First) та спочатку код (Code First). Підхід «спочатку код» дозволяє програмісту створювати класи сутностей із властивостями, а потім створювати БД і таблиці на основі визначених класів сутностей. Однак «спочатку БД» дозволяє створювати таблиці та базу даних, після чого модель даних сутності може бути створена за допомогою вже існуючої БД. Це і є основна відмінність між підходами «спочатку код» і «спочатку БД» у шаблоні проєктування MVC.

Утім використання DynamoDB накладає певні обмеження на те, яким чином можна проводити проєктування цієї БД. Оскільки дана БД є представником БД типу ключ-значення, вона має модель даних, що складається з пар ключ-значення в безсхемній, дуже великій, нереляційній таблиці рядків (записів). Відповідно,

звернення до даних відбувається шляхом доступу через головний ключ (primary key), в якості якого може виступати «ключ відділу» (partition key), або поєднання двох стовпців «ключ відділу» та «ключ сортування» (sort key). В такому випадку доступ до даних є дуже швидким, оскільки операція пошуку, вставки та видалення мають логарифмічну складність через використання бінарних дерев для зберігання даних. Утім коли постає задача видобутку даних з певними параметрами чи з певними залежностями, перед розробниками постає дилема: пожертвувати швидкістю операції чи пожертвувати пам’яттю.

DynamoDB передбачає операцію сканування, за допомогою якої можна використовувати параметри пошуку для співставлення запитуваних даних з даними з БД. Але в такому разі DynamoDB доводиться для кожної такої операції зчитувати абсолютно всі дані з таблиці, і вже після цього проводити порядкове порівняння для визначення відповідності того чи іншого рядка тим параметрам, які були вказані у пошуковому запиті. Само собою це максимально негативно відбивається на швидкодії процесів читання даних.

Інший спосіб полягає у використанні так званих глобальних другорядних індексів (global secondary index, GSI). Ця функція дозволяє створити для окремої таблиці її копію, в якій «ключ відділу» та «ключ сортування» будуть замінені на інші стовпці цієї таблиці. Наприклад, якщо є таблиця «Працівник», яка має стовпці «Id», «Підрозділ», «Ім’я», «Дата прийому на роботу», з яких перші два стовпці є ключем відділу та ключем сортування, відповідно, то для пошуку працівника, скажімо, за ім’ям доведеться використовувати операцію сканування. Але у разі використання GSI на стовпцях «Ім’я» та «Дата прийому на роботу», в БД буде умовно створено нову таблицю (а насправді ж буде створено нове дерево індексації до вже існуючих записів), яка буде мати в якості ключів відділу та сортування вже саме ці два стовпці, а стовпці «Id», «Підрозділ» будуть просто даними. В такому випадку швидкість операцій буде залишатися логарифмічною, але буде задіяно більше пам’яті для зберігання усіх дубльованих таблиць. А таких таблиць може бути не одні чи дві, а й ціла сотня в тому випадку, якщо записи

містять більше 100 стовпців, і по кожному з них необхідно окремо проводити пошук.

Оскільки однією з найголовніших причин використання NoSQL БД у проєктах є збільшення швидкодії зі збільшенням кількості записів та операцій в секунду, то очевидно, що варіант вибірки даних шляхом сканування не є оптимальним. А якщо використовувати варіант із застосуванням GSI, то постає необхідність розробки структури БД таким чином, щоб вона чітко відповідала запитам, які до неї будуть направлятися, щоб не отримати в кінці БД, яка займає занадто багато місця. А для цього починати розробку БД необхідно з визначення наперед усіх можливих запитів, які до неї будуть застосовані. Таким чином перед самою реалізацією БД чи то підходом «спочатку код», чи то підходом «спочатку БД», необхідно спроєктувати її дизайн.

3.6 Методи побудови траєкторії обходу покриття

У сучасному сільському господарстві використання технологій автоматизації та інформаційних систем є важливим етапом для підвищення ефективності обприскування сільськогосподарських полів. Одним із ключових елементів подібної інформаційної системи є алгоритм побудови траєкторії обходу покриття (ТОП) для БПЛА (дронів), які використовуються для розпилення рідких або порошкоподібних речовин на культури. Цей алгоритм визначає оптимальний маршрут руху дрона над полем, забезпечуючи ефективне покриття та оптимальне використання ресурсів.

Існує два основних способи розробки алгоритмів побудови ТОП: ручне програмування (методи прямолінійне, криволінійне планування, поділу на комірки тощо) та планування за допомогою розробленого ШІ та машинного навчання (а саме використання генеративних і адаптивних моделей). Алгоритми побудови ТОП можна програмувати вручну, або ж використати для цього ШІ.

Одним з методом побудови ТОП є використання ШІ або алгоритмів машинного навчання. В основі цього підходу лежить аналіз великого обсягу

даних та навчання моделі ШІ будуванню оптимальних траєкторій на основі вивчених патернів. Етапами цього методу є: збір та попередня обробка даних, розробка структури ШІ (якщо такої ще немає), навчання моделі та подальше використання моделі для побудови оптимальної траєкторії.

До переваг такого підходу можна віднести:

– автоматизація – використання ШІ дозволяє лише один раз розробити модель, яка після навчання буде автоматично підлаштовуватися під кожну нову область, і генерувати оптимальну ТОП для неї;

– адаптивність до змін – якщо модель навчена на хорошій виборці, то вона зможе якісно справлятися з побудовою ТОП для складних областей та адаптуватися до навколишніх умов;

– ефективне використання ресурсів – за правильного навчання моделі, вона зможе будувати траєкторію, яка буде оптимальною не тільки з точки зору часу її обходу, а й з точки зору економії зарядку дрона чи хімічної речовини.

З недоліків же метода можна зазначити:

– необхідність підготовки великої кількості даних – без об’ємної вибірки для навчання моделі не вийде отримати хороших результатів, а без попередньої обробки цих даних взагалі не факт, що вдасться навчити модель;

– складність розуміння – модель машинного навчання може бути складною для розуміння, а отже і налаштування;

– вплив якості даних на фінальний результат – окрім розміру навчальної вибірки немаловажливим фактором є її якість, без задовільного рівня якої модель не навчиться будувати оптимальної ТОП.

Інші способом розробки алгоритмів побудови ТОП є ручна реалізація. Цей спосіб є стандартним, і використовувався весь час до появи та розквіту ШІ. Принцип цього способу простий – для кожної окремої задачі розробляється підходящий алгоритм, який за думкою розробника повинен бути оптимальним. Для вирішення задачі побудови ТОП можна і не розробляти новий алгоритм, а просто адаптувати вже існуючий, або навіть поєднати декілька з них. Далі в цьому

розділі розглядатимуться різні методи побудови ТОП, алгоритми яких запрограмовані вручну.

Метод прямолінійного планування є одним з найбільш простих і ефективних способів планування руху дрона над ділянкою визначеною полігоном. Основна ідея полягає в тому, щоб створити прямі лінійні траєкторії, які будуть покривати більшу частину поля. Для цього спочатку визначаються границі поля, після чого визначена область розбивається на смуги, ширина яких зазвичай дорівнює ширині смуги обприскування дрона. Після цього вздовж обрахованих смуг прокладаються прямі лінії траєкторії.

Так, наприклад, у дослідженні [29] цей метод побудови ТОП називають бустрофедоном, який має схематичне зображення як на рисунку 3.8. Бустрофедон – це схема простого руху вперед і назад уздовж найдовшої сторони багатокутника Також в цьому дослідженні зазначається, що цей метод підходить для повного охоплення нескладної області без накладання.

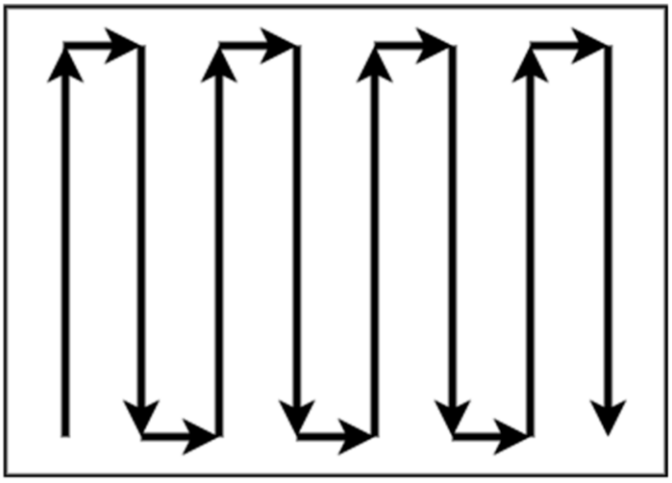

Рисунок 3.8 – Схема бустрофедона [29]

Цей метод має ряд переваг:

– простота й ефективність – прямі траєкторії є легкими для обчислення та реалізації, що робить їх ефективними для простих задач, особливо у випадках, коли геометрично поле нагадує форму прямокутника чи паралелограма;

– простий в реалізації – окрім того, що він вимагає мінімальних обчислювальних зусиль, він ще й доволі простий в плані кодування, оскільки не вимагає розробки та реалізації складних алгоритмів;

– ідеальний для рівних областей – якщо область, яка потребує побудови покриття, має форму наближену до простих геометричних фігур, то цей метод показує одні з найкращих результатів по продуктивності та швидкості.

Утім простота цього методу є одночасно і його найбільшим недоліком, що призводить до таких негативних проявів як:

– низька ефективність на нерівних областях – у разі великої нерівності поля (наявності великої кількості впуклих та опуклих і при цьому дуже гострих кутів) покрити поле неприривними лініями може бути майже неможливо, що призведе до значної сегментації траєкторії, що у свою чергу негативно вплине на продуктивність усього методу;

– обмежена адаптивність – метод може не враховувати і відповідно не підлаштовуватися до змін навколишнього середовища.

Схожим методом є метод криволінійного планування. Основна його ідея полягає в тому, щоб використовувати криві, які адаптуються до форми поля, забезпечуючи ефективне та рівномірне покриття. Прикладом такого методу побудови ТОП може бути квадратний патерн, який розглядається все у тому ж дослідженні [29] і має вигляд, зображений на рисунку 3.9.

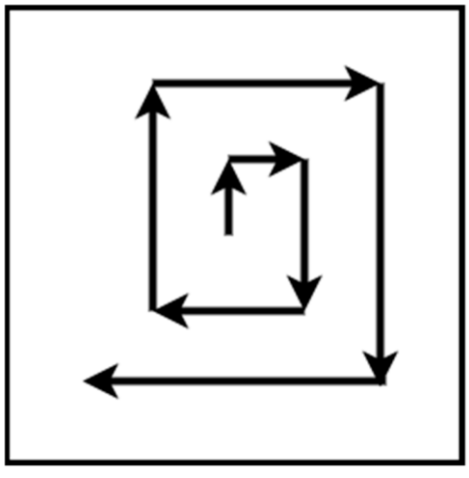

Рисунок 3.9 – Схема квадратного патерну покриття [29]

Цей метод будується за схожим алгоритмом, що і попередній, оскільки він включає ті ж етапи визначення границь поля, розбиття на смуги. Але також він включає додатковий етап – адаптацію до рельєфу, в якому відбувається процес огинання перешкод та оптимізація траєкторії.

До переваг такого методу можна віднести:

– адаптивність – метод підлаштовується під рельєф, за рахунок чого значно зменшується площа непокритих ділянок поля;

– рівномірне обприскування – за рахунок зменшення площі непокритої області збільшується відсоток обприсканих посівів, а за рахунок огинання перешкод замість розриву траєкторії збільшується рівномірність нанесення хімічних речовин;

– ефективність на нерівних полях – метод є непоганим варіантом для нерівних областей, які можна покрити за рахунок, наприклад, обходу їхнього периметру з внутрішньої сторони поступово збільшуючи відступ з кожним таким обходом.

Але метод має і свої недоліки:

– складність обчислення – метод може бути як не дуже складним, так і досить затратним у плані ресурсів, що залежить від потрібної точності покриття і, відповідно, складності алгоритму;

– складність реалізації – метод є складнішим в реалізації, оскільки алгоритм повинен враховувати значно більше умов при побудові траєкторії, що займає більше часу та зусиль розробника;

– вплив на ефективність дрона – в залежності від кривизни траєкторії та інших її параметрів може змінюватися і швидкість роботи дрона, що призведе до зменшення його ефективності.

Також у дослідженні [29] виділяють метод клітинної декомпозиції, який поділяє нерегулярний простір на клітини (комірки). Цей метод у свою чергу поділяється на два типи: точної та наближеної декомпозиції. Методи точної декомпозиції розкладають область на частини, з’єднання яких гарантує утворення точної копії оригінальної області. В той час як методи наближеної декомпозиції

розбивають область на частини, якими легше оперувати, але в такому разі втрачається деяка точність у відсотку покриття оригінальної області.

До точних декомпозиційних методів можна віднести метод трапецієвидної декомпозиції. Кожна клітинка в цьому методі є трапецією, і тому для її обходу можна використовувати прості рухи, такі як вперед і назад. Покриття може бути досягнуто шляхом простого лінійного обходу, як той же метод бустрофедону, як показано на рисунку 3.10.

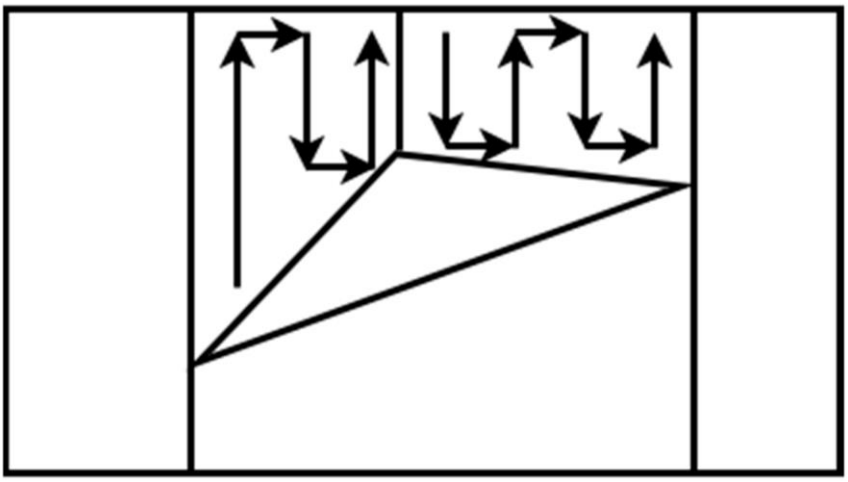

Рисунок 3.10 – Схема трапецієвидної декомпозиції [29]

Розширенням трапецієвидної декомпозиції може бути бустрофедонська декомпозиція. Цей вид декомпозиції формується через поєднання трапецієвидних клітинок з попереднього методу таким чином, щоб новоутворені клітини можна було обійти за допомогою бустрофедону (рис. 3.11). Таким чином відбувається зменшення загального числа клітин, що у свою чергу зменшує загальну довжину ТОП, що у свою чергу зменшує час обприскування та економить заряд дрона.

Ще одним цікавим методом точної декомпозиції є метод розкладання за допомогою зрізів. Він працює наступним чином: для визначення меж клітинок декомпозиції використовується пряма лінія, що перетинає усю область і відображає бажане направлення вздовж якого буде відбуватися обхід, і яка рухається з одного краю області в інший.

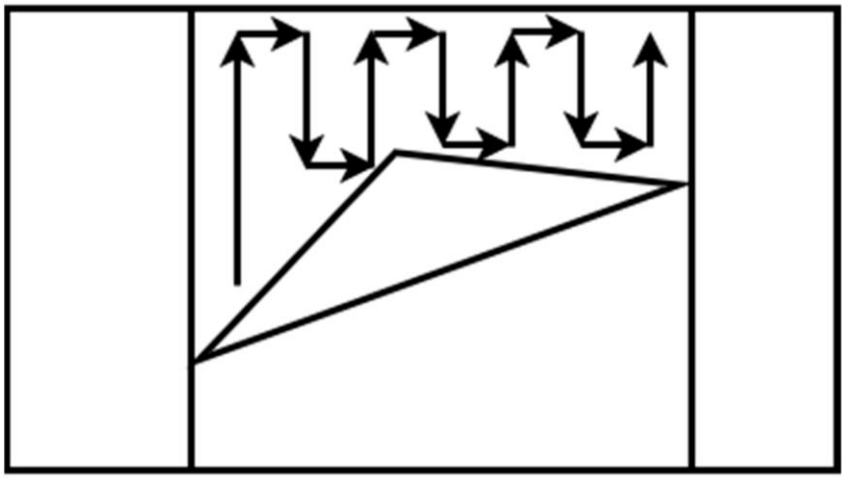

Рисунок 3.11 – Схема бустрофедонської декомпозиції [29]

Коли лінія проходить по області, дійшовши до перешкоди, вона розділяє цю область на дві або більше клітинок (рис. 3.12). Після чого для кожної клітинки вже можна будувати свою ТОП, наприклад, тим же бустрофедоном.

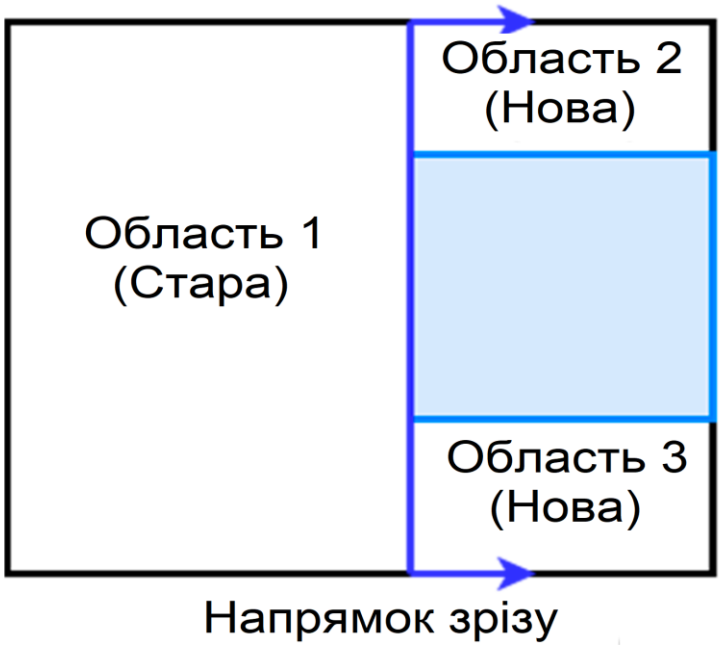

Рисунок 3.12 – Схема декомпозиції за допомогою зрізів [29]

Але окрім точних методів декомпозиції є ще й наближені. Так наприклад, методи декомпозиції, які використовують сітку для розкладання області, можна віднести до цієї категорії. Оскільки клітинки сітки є чітко визначеного розміру, з їх допомогою неможливо покрити усі 100% ориганльної області рівномірно

(тобто без накладання). Клітинки сітки найчастіше мають форму квадрати, але можуть бути й іншої форми, як от шести- чи восьмикутниками. Приклад розкладання за допомогою сітки можна побачити на рисунку 3.13.

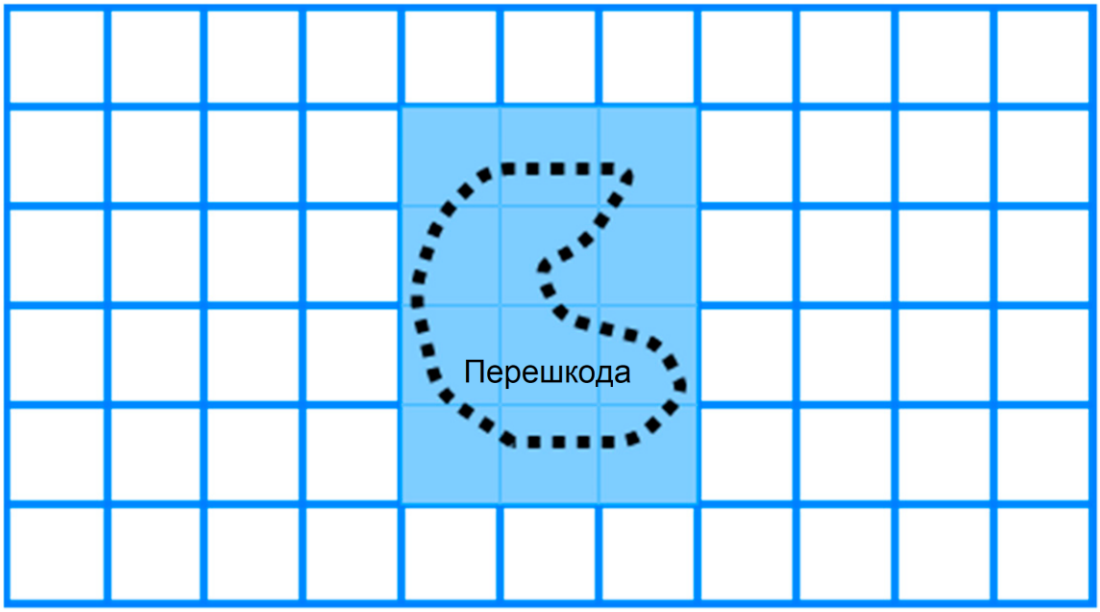

Рисунок 3.13 – Приклад декомпозиції за допомогою сітки квадратів [29]

Цей метод зазвичай використовують у випадку, коли область, яка потребує побудови ТОП, дуже нерівна, та (або) коли присутні перешкоди, з дуже нерівною геометрією. У тому чи іншому випадку при спробі декомпозиції точними методами буде утворено безліч маленьких і вузьких клітинок, що максимально негативно відіб’ється на швидкості обходу траєкторії, і як результат на заряді дрона та часі роботи загалом. Для того аби зменшити кількість клітинок, і використовуються наближені методи декомпозиції.

Окрім алгоритмів декомпозиції також слід розглянути й алгоритми знаходження найкоротшого шляху, оскільки під час побудови траєкторії можуть трапитися ситуації, коли траєкторія була побудована таким чином, що неохоплені клітинки ще залишилися, а усі сусідні клітинки поряд з поточною позицією вже було пройдені. В такому випадку необхідно перемістити поточну позицію до найближчої непройденої клітинки.

Існує багато алгоритмів пошуку найкоротшого шляху та їх модифікацій. Найбільш простими і за сумісництвом найбільш популярними є алгоритми пошуку спочатку вшир та спочатку вглиб, алгоритм Дейкстри та A* (чит. «а стар»).

Пошук спочатку в ширину (breadth first search, BFS) є одним із двох найбільш фундаментальних алгоритмів обходу графів. Він гарантує найкоротший шлях між двома вузлами, у випадку коли з'єднання мають однакову вартість. Пошук у ширину досліджує в усіх напрямках однаково, не використовуючи ніяких допоміжних параметрів, доки не буде досягнуто мети. Іншими словами, він починає з вибраного вузла та перевіряє всіх його сусідів, потім перевіряє всіх сусідів сусідів і так далі.

Пошук спочатку вглиб (depth first search, DFS) працює за схожим принципом, але замість того, аби спочатку перевірити усі сусідні вузли, він робить обхід спочатку по одній гілці, допоки не зустріне тупик або не досягне мети. Якщо мети не досягнуто, він повертається на один рівень назад, і перевіряє сусідню гілку до її закінчення. І таким чином він продовжує діяти допоки не буде досягнуто кінцевої мети.

Обидва ці алгоритми дають повну гарантію того, що кінцева бажана позиція буде знайдена, але вони витрачають занадто багато кроків для її знаходження, оскільки ніяким чином не корегують свою поведінку під час роботи. Окрім того, якщо пошук вшир ще теоретично можна застосовувати у випадку знаходження найближчої клітинки на двовимірній площині, то пошук вглиб буде найповільнішим з усіх. Це пов'язано з тим, що непройдена клітинка найімовірніше не буде розташована дуже далеко від поточної позиції, а пошук вглиб може вибрати для перевірки гілку, яка розташована найдалі від неї. І тоді навіть якщо найближча непройдена клітинка буде розташована на відстані 2-3 клітинки вліво від поточної позиції, то допоки пошук вглиб не перебере усі можливі варіанти справа, він не підійде до розгляду цього варіанту.

Алгоритм Дейкстри у свою чергу вже є більш просунутим, бо він вже використовує евристики для оптимального пошуку. Алгоритм Дейкстри

знаходить найкоротший шлях від кореневого вузла до кожного іншого вузла, допоки не буде досягнуто цілі. Він є одним із найкорисніших графових алгоритмів; крім того, його можна легко модифікувати для вирішення багатьох різних проблем.

І останній у списку – алгоритм A*. Він є покращеною версією алгоритму Дейкстри, і може знаходити найкоротший шлях швидше за усі інші перелічені алгоритми. Через свою простоту і ефективність цей алгоритм знайшов найбільшу популярність. В той час коли алгоритм Дейкстри працює ненайбільш ефективно, витрачаючи час на дослідження в напрямках, які не є перспективними, A* дозволяє включати додаткову інформацію, яку алгоритм може використовувати як частину евристичної функції задля покращення ефективності. Коли алгоритм Дейкстри використовує тільки відстань від кореневого вузла, то алгоритм A* використовує як фактичну відстань від кореневого вузла, так і розрахункову відстань до мети від поточної позиції.

Висновки до розділу 3

В цьому розділі було проведено аналіз різних типів архітектури системи, розглянуто їхні переваги та недоліки, після чого обрано один з типів для його реалізації в системі, що розробляється. Цією архітектурою стала трирівнева клієнт-серверна архітектура.

Також в цьому розділі було визначено функціональну складову системи за допомогою акторів та прецедентів. Таким чином з’ясовано основі функції, які має виконувати система, та з якими буде взаємодіяти користувач. Після цього було сформульовано порядок взаємодії користувача із системою, який знайшов своє відображення у діаграмі активності.

Наступним кроком було визначено засоби розробки системи. Визначено, що через ряд зазначених переваг оптимальним рішенням буде використати платформу ASP.NET Core для побудови основи системи. Також з цією ціллю буде використано шаблон проєктування MVC. Для взаємодії користувача з картами

буде використано OpenLayers API, який зазвичай працює на платформі Node.js. Для того щоб мати можливість використовувати API без необхідності запускання іншої платформи, обрано використати пакувальник Webpack, який сформує з Node.js модулів та JavaScript файлу, в якому вони викликаються, окремий файл-збірку, яка може працювати автономно. А для обрахунку ТОП буде використано модуль, написаний на мові Python. Використання цієї мови зумовлене легшою розробкою та легшим усуненням помилок, а також наявністю більшої кількості користувацьких бібліотек з допоміжним функціоналом.

Ще в цьому розділі було обрано БД, в якій зберігатимуться дані системи, та розроблено її структуру. Цією БД стала DynamoDB. А її структура містить декілька таблиць, які окрім самих даних містять ще й посилання на головні ключі пов’язаних таблиць для реалізації їх зв’язків.

І на кінець було розглянуто алгоритми, які можна використати під час процесу побудови траєкторії обходу покриття. Так для того аби мати змогу побудувати траєкторію було розглянуто методи обходу за бустрофедоном та за квадратом, методи декомпозиції трапецієвидної, за допомогою бустрофедону та зрізів, а також метод декомпозиції сіткою. Також розглянуто алгоритми знаходження найближчого непройденого вузла.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Опис структури системи

Раніше в цій роботі було визначено, що найкращим варіантом для реалізації системи буде використати клієнт-серверну архітектуру, а саме її підвид – трирівнева архітектура. Також було обрано шаблон MVC для побудови цієї архітектури. Відповідно до цих рішень було розроблено структуру системи, яку в узагальненому вигляді можна побачити на рисунку 4.1, а в розширеному в додатку Г.

Рисунок 4.1 – Спрощена структурна схема системи

В цій структурі компонент «Клієнт» є узагальненим відображенням усіх клієнтів системи. Клієнт має у своєму розпорядженні графічний веб інтерфейс та з його допомогою здійснює взаємодію із системою. З функцій, які йому доступні, є реєстрація й авторизація, взаємодія з картою, додавання та видалення РО та ТЗ,

додавання, редагування, видалення дронів та їхніх моделей, додавання, редагування та видалення РП, а також їхній запуск та моніторинг процесу їх виконання. Детальніше про вигляд клієнта та його функціонал дивись у підрозділі 4.3.

4.1.1 API контролери

Усі клієнти є віддаленими, запускаються через браузер на пристроях користувачів і зв’язуються з сервером через мережу Інтернет. Взаємодія із системою відбувається через API контролери, які обробляють HTTP запити від клієнтів та відправляють їм відповіді. В контролерах зосереджена основна частина бізнес-логіки системи. Побачити увесь перелік функцій, які мають контролери можна на рисунку 4.2.

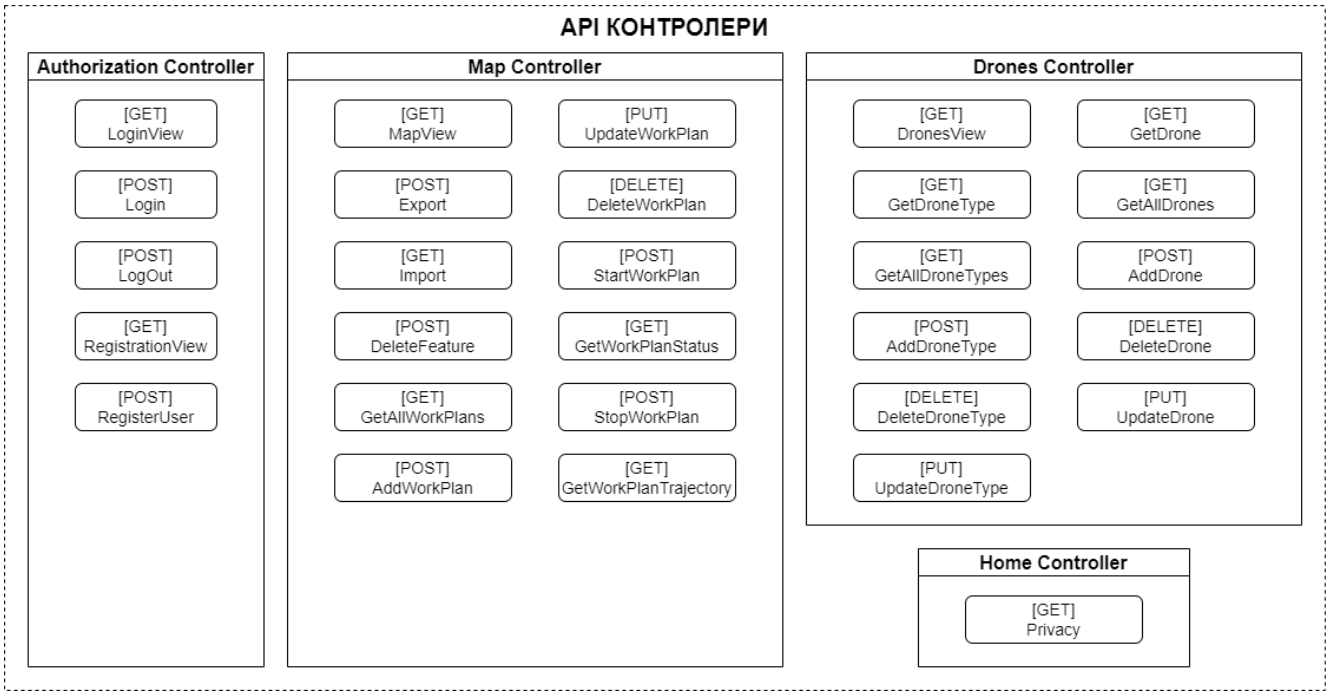

Рисунок 4.2 – Контролери та їх функціонал

Першим у списку йде контролер авторизації (Authorization controller). Він же є першим, з яким взаємодіє користувач під час входу в систему. Його зона відповідальності – обробка запитів на реєстрацію нових акаунтів та входу

(виходу) в (з) систему (-и) вже зареєстрованих. Також він відповідає за надання користувачам представлень (веб сторінок з відповідним наповненням), через які й буде відбуватися взаємодія користувачів із цим контролером. Ними є представлення Авторизації та Реєстрації.

Наступний з переліку – контролер карти (Map controller). Він має найбільше функціоналу з усіх контролерів. Його зона відповідальності – обробка запитів на керування об’єктами карти, керування РП, їх запуск, зупинка, моніторинг виконання, а також видобування ТОП. Як і попередній контролер він надає користувачу представлення карти. Функції Export, Import, DeleteFeature відповідають за збереження, видобуток та видалення об’єктів карти (РО та ТЗ) до (з) БД, відповідно. Таким самим чином працюють і наступні чотири функції, які відповідають за зчитування, запис, редагування та видалення інформації про РП. Функції StartWorkPlan та StopWorkPlan, відповідно, відповідають за запуст та зупинку обраного користувачем РП. GetWorkPlanStatus відповідає за моніторинг процесу виконання РП, надсилаючи користувачу поточну позицію дрона, його поточну швидкість, наповненість резервуара, заряд батареї тощо. Функція GetWorkPlanTrajectory відповідає за відправку користувачеві координат ТОП для поточно обраного ним РП, який буде відображений на його карті в представленні.

Ще одним важливим контролером є контролер дронів (Drones controller). Він відповідає за постачання користувачеві представлення для керування дронами та їх моделями. Функції GetDrone та GetDroneType цього контролера направлені на те, щоб надати користувачу інформацію про дрон чи його модель, яку він запитує в сервера. У свою чергу функції GetAllDrones та GetAllDroneTypes надають інформацію про усі наявні на підприємстві дрони або їх моделі. Також цей контролер здійснює додавання, редагування та видалення інформації про дрони та їх моделі за допомогою відповідних функцій.

Останнім серед контролерів є домашній контролер (Home controller), який відповідає за усі базові функції, загальні для усієї системи. Наразі він містить лише функцію, яка надає користувачам представлення з інформацією про політику конфіденційності.

Усі контролери так чи інакше але мають в наявності хоча б одну функцію, яка надає користувачам представлення, яке містить інтерфейс для взаємодії з функціоналом цього контролера. Перелік представлень можна побачити у структурній схемі системи в додатку Г. Детальний розгляд інтерфейсів, які надають користувачам представлення, здійснений у підрозділі 4.3.

4.1.2 Сервіси

Окрім бізнес-логіки, яка розміщена в контролерах, є ще й та, яка була винесена в окремі модулі – сервіси. Основним мотивом такого кроку є те, що ці модулі спроєктовані таким чином, щоб мати можливість застосовуватись універсально – без прив’язки до контексту певного контролера. Побачити повний перелік функцій, доступний через сервіси, можна на рисунку 4.3.

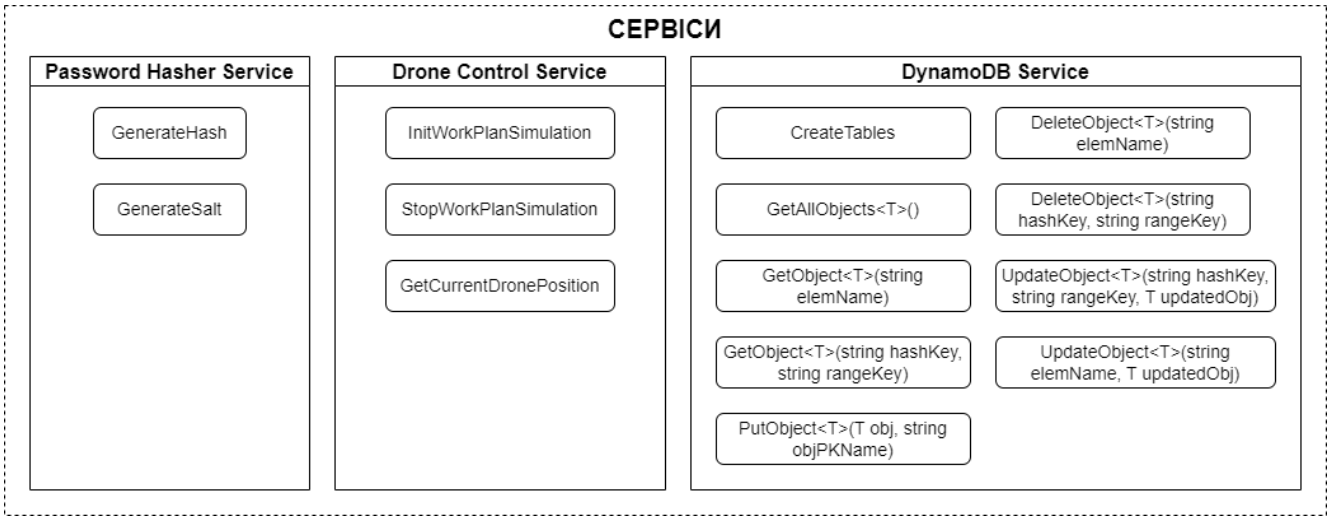

Рисунок 4.3 – Сервіси та їх функціонал

Першим у списку сервісів іде сервіс хешування паролів (password hasher service). Він має дві функції – генерація хешу пароля (password hash) та генерація солі (password salt).

Оскільки зберігання пароля у незміненому вигляді у БД (чи будь-яким іншим чином) є прямим порушенням безпеки, необхідно впевнитися, що пароль надійно приховано. Але простого зберігання пароля у захищеній паролем БД чи

звичайного шифрування пароля недостатньо для забезпечення повної безпеки. Перший варіант надає умовну безпеку, але зловмисникам варто зламати лише один пароль (пароль від БД) і всі паролі (як і вся інша інформація) користувачів будуть в ту ж мить скомпрометовані. Другий варіант так само не дає хорошого захисту, оскільки шифрування є двонапрямленим методом – інформацію можна як зашифрувати, так і розшифрувати назад у початковий вигляд. Тому після отримання шифрувальної функції зловмисник зможе з легкістю дешифрувати усі паролі.

Щоб подібних недоліків уникнути існує один доволі дієвий метод – хешування. Хешування на відміну від шифрування є однонапрямленим процесом – тобто хешувавши пароль, отримати його назад із згенерованого хешу вже буде дуже складно, а в деяких випадках майже неможливо. В такому випадку принцип роботи з паролями буде наступний:

– користувач вводить у спеціальне поле форми свій пароль;

– по затвердженню форми пароль надсилається на сервер у первозданному вигляді захищеним шляхом (використовуючи HTTPS протокол);

– сервер, використовуючи функцію хешування, отримує хеш пароля та зберігає його у відповідному полі в БД;

– при наступному вході користувача в систему, відбувається весь той самий процес, але замість збереження хешу пароля сервер порівнює хеш, який був згенерований з надісланого користувачем пароля, та хеш, який збережений за користувачем в БД, і в разі співпадіння, вхід в систему дозволяється.

Але попри захищеність такого методу, він має й слабкість – прості, короткі та тривіальні паролі генерують простий хеш, у випадку якщо використовувати просту хеш-функцію. Для того, що уникнути цього, зазвичай використовують вже завчасно розроблені та відомі хеш-функції, які є досить складними, та які генерують доволі довгий хеш. Найбільш розповсюдженою є функція SHA256, яка генерує хеш у розмірі 256 байтів.

Та навіть за варіанту, коли довжина згенерованого хешу є сталою, одначе залишається велика вірогідність того, що за використання користувачами

однакового простого та тривіального пароля, буде згенеровано однаковий хеш. Тому для того щоб нівелювати ймовірність такого сценарію, було розроблено метод підсолювання та перцювання (salting and peppering).

Сіль (salt) є випадково згенерованим набором байтів, який генерується окремо для кожного користувача під час першої його реєстрації. У незміненому вигляді цей набір байтів зберігається у відповідне поле запису цього користувача в БД. Після чого сіль додається до згенерованого від паролю хешу кожен раз при створенні, зміні чи перевірці пароля. Таким чином навіть за наявності однакового паролю, у двох різних користувачів буде виходити різний кінцевий хеш, якщо тільки випадкова сіль не вийшла однаковою в них обох. Але вірогідність збігу такого числа обставин прямує до нуля.

Перець (pepper) у свою чергу працює подібно до солі, але з тією відмінністю, що він є не випадково згенерованим, а завчасно запрограмованим, статичним та однаковим для усіх користувачів. Хорошою практикою є зберігання цього набору байтів у місці, до якого не буде прямого доступу з мережі (файл налаштування проєкту для ASP.NET застосунків, або змінна середовища записана неопосередковано у коді), а також наявність певного набору перців, з їх періодичною заміною.

Також для ще більшого захисту пароля можна використати подвійне або потрійне хешування. Робиться це для того, аби сіль та перець мали ще більший вплив на фінальний результат. В такому разі можна зациклити функцію хешування саму на собі на певну кількість ітерацій: в першу ітерацію як параметри у функцію передаються рядки пароля, солі та перцю, а також кількість ітерацій; після хешування відбувається рекурсивний виклик цієї ж функції, але замість рядка пароля вже подається рядок хешу, а число ітерацій зменшується на одиницю; по закінченню числа ітерацій буде отримано дуже міцний хеш.

Повну схему алгоритму хешування пароля можна побачити на рисунку 4.4.

Окрім сервісу хешування паролів система також має у своєму розпорядженні сервіс контролю дронами. Цей сервіс має функції запуску та зупинки виконання РП, а також функцію моніторингу їх прогресу виконання.

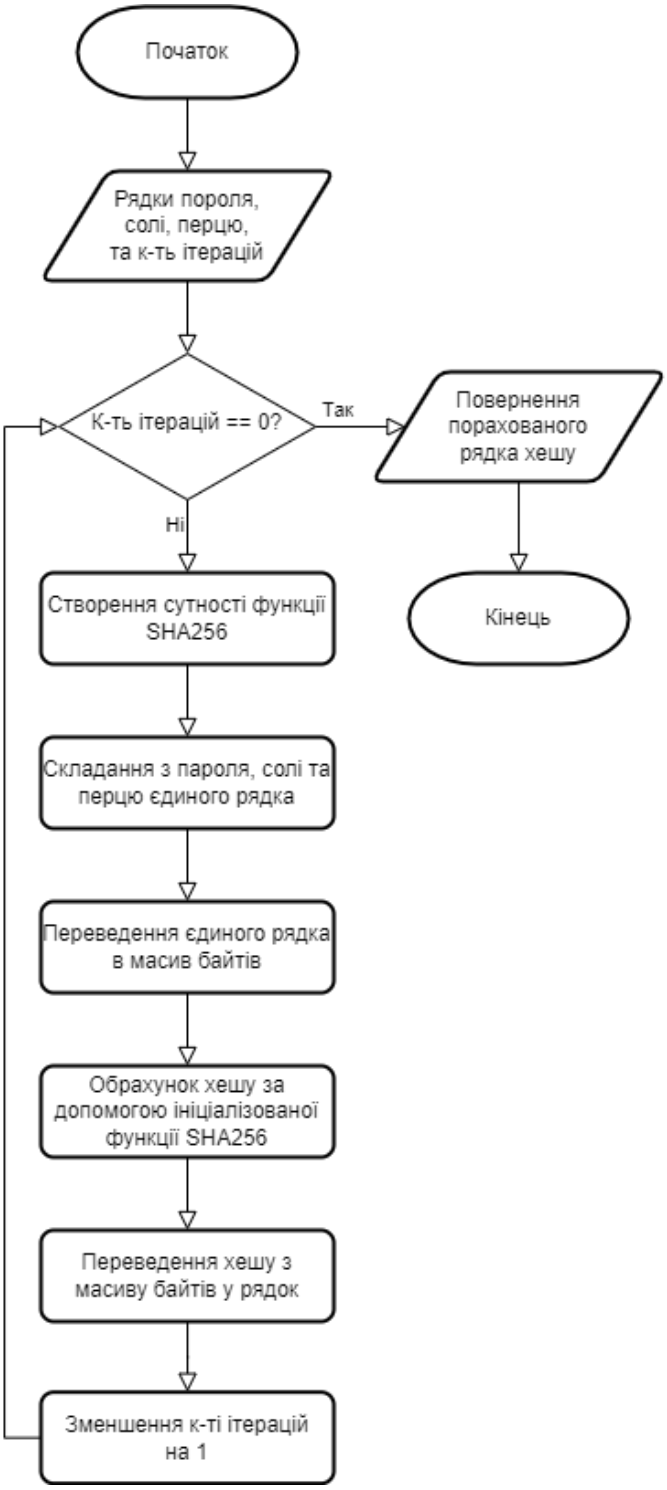

Рисунок 4.4 – Алгоритм хешування пароля

Функція InitWorkPlanSimulation проводить ініціалізацію початку виконання РП. Вона створює сутність моделі DroneControl, яка зберігає поточні параметри РП та виставляє час початку роботи.

Функція GetCurrentDronePosition повертає поточну позицію дрона, для відображення його на карті у користувача.

Функція StopWorkPlanSimulation зупиняє виконання РП або по виконанню поставленого завдання з обприскування, або при отриманні запиту зупинки роботи від клієнта.

І останнім сервісом в системі є сервіс роботи з DymaoDB. Оскільки під час роботи системі необхідно постійно робити запити до БД, дуже помічним рішенням є винесення функціоналу в окремий модуль, до якого будуть звертатися контролери у разі необхідності взаємодії з БД. Це дозволить спростити не тільки процес розробки, а й процес виявлення та вирішення помилок, додавання нового функціоналу та редагування поведінки старого. За наявності функціоналу, в якому операції визначені за допомогою генеративних об’єктів, можна не турбуватися про дублювання коду (що є слідуванням принципу DRY – don’t repeat yourself, не повторюй себе) або про розподіленість логічно пов’язаного функціоналу між різними контролерами (що є слідуванням принципу Single responsibility principle – принцип єдиної відповідальності).

Таким чином сервіс має функції створення таблиць в БД, видобуток об’єктів з неї, додавання, видалення та редагування об’єктів у БД. Так, наприклад, функція створення таблиць в БД виконується під час запуску усієї системи, що є реалізацією підходу «спочатку код» в проєктуванні БД. Це дозволяє розробити функціонал системи спираючись на моделі, а вже під час першого запуску на основі моделей буде створено відповідні їм таблиці в БД. Тим паче, цей процес проходить значно простіше під час використання DynamoDB, оскільки ця БД не має звичних зв’язків між таблицями, що значно спрощує їх проєктування через кодування моделей. Окрім цього більшість функцій в цьому сервісі є перевантаженими, що дозволяє звертатися як до таблиць, які мають в якості головного ключа лише хеш-ключ, так і до таблиць, у яких головний ключ представлений комбінацією хеш-ключа та ключа сортування.

4.1.3 Моделі

Як можна побачити на додатку Г, система має дев’ять моделей, серед них: користувач (User), РО (Area), ТЗ (Point), РП (Work Plan), ТОП (Coverage Trajectory), дрон (Drone), модель дрона (Drone Model), DroneControl та CmdRun.

Кожна з моделей репрезентує певний клас, який може містити інформацію та (або) певний функціонал. Так, наприклад, моделі користувача, РО, ТЗ, РП, ТОП, дрона та моделі дрона є класами, які тільки зберігають певну інформацію. Вони мають поля, які є відображенням колонок таблиць у БД. Саме з цих класів формуються таблиці БД під час першого запуску системи, а також з їх допомогою здійснюється взаємодія з ними.

У свою чергу класи DroneControl та CmdRun мають трохи інше призначення. Перший містить інформацію про поточний прогрес виконання РП, який перевіряється під час його роботи. Другий містить функціонал запуску виконуваних файлів Python, що необхідно для роботи з модулем побудови ТОП.

4.1.4 База даних

Для того, щоб побудувати БД, найпершим ділом необхідно визначити усі сутності, які будуть присутні в системі, та які зберігатимуть дані в таблицях БД. Для цього сформуємо перелік усіх сутностей та їх опис (див. табл. 4.1) на основі діаграми прецедентів, яка була представлена раніше. Ці сутності також є відображенням моделей, які присутні в системі.

Таблиця 4.1 – Перелік та опис сутностей БД

Сутність	Опис
Працівник	Містить усю інформацію пов’язану з працівником певного підприємства, зокрема: ім’я користувача, «сіль пароля» (password salt) та хеш пароля, посилання на підприємство, в якому працює.

Сутність	Опис
Підприємство	Містить усю інформацію пов’язану з підприємством, зокрема: назву підприємства, юридичну адресу, корпоративну електронну пошту та телефон.
Дрон	Містить усю інформацію пов’язану з певним дроном, який доступний на підприємстві, зокрема: унікальну назву, посилання на підприємство, якому належить, та на модель дрона.
Модель дрона	Містить усю інформацію пов’язану з моделлю дрона, зокрема: унікальну назву, об’єм резервуара, мінімальну та максимальну ширину смуги обприскування, мінімальну та максимальну витрату речовини та максимальну швидкість.
Робоча область	Містить усю інформацію пов’язану з робочою областю, яка створена підприємством, зокрема: унікальну назву області, координати її полігона та посилання на підприємство, яке її створило.
Точка завантаження	Містить усю інформацію пов’язану з точкою завантаження, яка створена підприємством, зокрема: унікальну назву точки, її координати та посилання на підприємство, яке її створило.
Траєкторія покриття	Містить усю інформацію пов’язану з ТОП, яка створена підприємством та закріплена за певною комбінацією РО та ТЗ, зокрема: унікальну назву області, координати ТОП, посилання на РО, ТЗ та на підприємство, яке її створило.

Сутність	Опис
Робочий план	Містить усю інформацію пов’язану з робочим планом, який акумулює в собі параметри роботи, зокрема: унікальну назву, посилання на РО, ТЗ, дрон та ТОП, які використовуються в ньому, а також швидкість, ширину смуги обприскування та витрати речовини дрона, на який посилається.

Враховуючи вказані сутності та їхній опис, можна побудувати діаграму залежностей (ER діаграма). ER діаграма візуально пояснює зв’язок між сутностями, які будуть записані в БД. ER діаграма – це, по суті, структурний дизайн БД. Він служить основою зі спеціальними символами, призначеними для визначення того, як сутності бази даних пов’язані між собою. Три основні елементи служать основою для створення діаграми ER: зв’язки, сутності та характеристики.

Побудовану ER діаграму для системи, що розробляється в цій роботі, можна побачити у додатку Д.

Але, слід розуміти, що DynamoDB це нереляційна БД, і тому прямих зв’язки між таблицями побудувати неможливо. Отже, необхідно вигадати спосіб, як можна пов’язати сутності між собою іншим способом. Таке завдання в DynamoDB можна вирішити двома способами:

– додати до кожної таблиці стовпчик з посиланнями на головні ключі усіх залежних таблиць;

– зберігати дані усіх сутностей в одній таблиці, при цьому для доступу до необхідних даних використовувати GSI.

Перший варіант є найпростішим, оскільки він не порушує нормалізації даних, і простіший у впровадження. А також за цього варіанту БД можна буде легко масштабувати, додавши нові таблиці, якщо з’явиться така необхідність. В той час як при однотабличній архітектурі БД (single table architecture), зробити це

вдасться лише за перебудови усієї таблиці, що по суті означає перебудову усієї БД.

Отже, в такому разі для побудови структури БД системи, що розробляється, буде використаний перший спосіб. Тоді її фізична модель буде виглядати як набір незалежних таблиць, в кожній з яких буде окремо виділений стовпець для посилання на необхідний рядок з іншої залежної таблиці. В той час як в реляційній БД необхідно було б використовувати JOIN команди для того, щоб сформувати відповідь, яка б включала запитувані дані з усіх пов’язаних таблиць, у випадку з DynamoDB буде робитися декілька запитів для отримання усієї необхідної інформації.

Враховуючи розроблену БД та структуру системи, розроблено відповідні моделі та контролери, класи яких зображені в додатку Е.

4.2 Реалізація алгоритму побудови траєкторії обходу покриття

Як вже зазначалося у 3.6 перед побудовою траєкторії необхідно провести сегментацію (декомпозицію) області, щоб отримати найбільш прийнятний для поставлених задач шлях. Також там було розглянуто різні алгоритми декомпозиції, які допомагають в цій задачі. Але аби побудувати оптимальну ТОП для визначеної області не достатньо зробити лише її декомпозицію і обійти її за допомогою бустрофедону. Для того аби знайти оптимальний обхід побудованих клітинок необхідно спроєктувати алгоритм, який буде будувати ТОП таким чином, щоб обхід відбувся у найменшу кількість кроків і в той же час з найбільшим відсотком покриття.

Для початку треба визначитися з методом декомпозиції області. Оскільки раніше вже було зазначено, що у разі використання точних методів за наявності великої нерівності самої області чи перешкод всередині неї кількість клітинок розкладання може бути дуже великою, а самі клітинки можуть бути різної форми (товсті, тонкі, витягнуті, короткі), то очевидним рішенням буде вибрати декомпозицію за допомогою сітки. Це рішення може і зменшить відсоток

загальної площі покриття, але значно спростить розробку алгоритму, а також позитивно відобразиться на загальній довжині траєкторії та часі роботи. Задля простоти розробки було обрано розбиття на клітинки квадратної форми.

Тепер потрібно зрозуміти, яким чином спроєктувати алгоритм обходу клітинок, щоб отримати найкоротший шлях. Для вирішення цієї задачі було обрано застосувати такий метод – використовуючи чотири типи евристичних функцій для визначення відстані між клітинками, побудувати чотири траєкторії покриття, і з них вибрати ту, в якій фінальна кількість кроків буде найменша. Таким чином буде досягнуто оптимальності у параметрі довжини траєкторії.

Перелік евристичних функцій, які будуть використані в алгоритмі, можна побачити на рисунку 4.5.

Рисунок 4.5 – Ілюстрація роботи евристичних функцій

Першою з них є манхетенська евристична функція, яка використовує махетенську відстань для визначення відстані між клітинками. Махетенська відстань у свою чергу є ні чим іншим як простим середнім значенням різниць по координатах. Наступною буде використана евристична функція Чебишева, особливістю якої є те, що відстань до сусідніх клітинок в усіх напрямках (по горизонталі, вертикалі та діагоналі) є однаковою. Таким чином рух заохочується в усіх напрямках рівносильно. Третьою та четвертою евристичними функціями є горизонтальна та вертикальна. Вони влаштовані таким чином, що відстань до клітинок, які лежать на одній горизонталі (вертикалі) з вихідною точкою вважається нульовою, а відстані до точок, що лежать на віддалених від центру горизонталях (вертикалях), збільшується на 1. Таким чином рух заохочується найбільше саме в горизонтальному або вертикальному напрямках.

Оскільки алгоритм A* є найбільш ефективним серед усіх чотирьох розглянутих у підрозділі 3.6, саме його і буде використано для пошуку найближчої непройденої клітинки.

Узагальнену блок-схему процесу побудови ТОП можна побачити у додатку Ж. Процес починається з того, що до нього поступають координати полігона РО, координати ТЗ та радіус смуги обприскування дрона. Далі РО розбивається на сітку з квадратів, довжини сторін яких рівні діаметру смуги обприскування дрона (або ж двом радіусам). Потім в процесі задаються 4 евристики, по яким проводиться перебір з побудовою траєкторії для кожної з них. Траєкторія будується по координатах, які є центрами клітинок сітки. Якщо траєкторія побудована, то процес переходить до наступної евристики. Якщо ж в процесі побудови виникла ситуація відсутності вільної клітинки поруч з поточною позицією, коли ще не всі клітинки пройдені, то викликається функція знаходження найближчої вільної клітинки. Якщо така клітинка знайдена, побудова ТОП продовжується від неї. Якщо ж ні, то процес побудови закінчується і відбувається перехід до наступної евристики. Коли усі евристики пройдені з усіх побудованих траєкторій вибирається та, яка має найменшу кількість кроків, і виводить в якості відповіді.

Алгоритм побудови сітки для РО можна побачити у додатку И. Першим ділом проводиться побудова так званого «внутрішнього полігона» – полігона, який буде вписаним у полігон вихідної РО та всі точки якого будуть рівновіддаленими від точок полігону вихідної РО на відстань рівну радіусу ширини смуги обприскування. Потім координати побудованого внутрішнього полігона разом з координатами ТЗ та радіусом ширини смуги обприскування передаються далі в алгоритмі. Далі з координат внутрішнього полігона беруться найбільші та найменші значення по осях Х та У, які формують межову рамку (boundary box, bbox). Далі в подвійному циклі визначаються центри квадратних клітинок сітки. Клітинки мають розмір рівний довжині смуги обприскування. Якщо центр клітинки потрапляє в межі внутрішнього полігона, то ця клітинка додається до матриці сітки з позначкою 0, в іншому ж випадку – з позначкою 1. Подібні позначки необхідні для алгоритму побудови ТОП для визначення доступних для обходу клітинок. Так позначка 0 означає, що клітинка доступна для обходу, позначка 1 – перешкоду, позначка 2 – початок побудови траєкторії.

Після того як матрицю сітки, яка складається з координат центрів клітинок з їх позначками, було побудовано, проводиться операція з обертання матриці на 90° для того, аби поміняти місцями рядки зі стовпчиками, для коректного відображення сітки. Після розвороту матриці сітки відбувається пошук клітинки, з якої почнеться побудова траєкторії. Для цього проводиться перебір усіх доступних для обходу клітинок з перевіркою їхньої відстані до ТЗ. І тій клітинці, що є найближчою до ТЗ, призначається позначка 2 – початок траєкторії.

На кінець алгоритму проводиться додавання до матриці сітки додаткового рядка, який є набором координат точок внутрішнього полігона. Цей рядок не буде брати участі у побудові оптимальної ТОП, але після побудови такої буде доданий на кінець траєкторії. Таким чином ТОП буде завершуватись обходом РО по периметру. Це зроблено для того аби нівелювати усі пропущені ділянки, що розташовані близько до границь РО, та які можуть часто випадати з сітки через те, що метод декомпозиції сіткою є приблизним, а не точним. Фінальна матриця сітки з додатковим рядком відправляється далі в алгоритм побудови ТОП.

4.3 Реалізація інтерфейсів користувача

На ряду з реалізацією функціоналу системи також стоїть і створення інтерфейсів користувача. Вони є невід'ємною частиною роботи із системою, а також вони є першою річчю, з якою стикається і взаємодіє клієнт. Тому дуже важливо спроєктувати такий інтерфейс, який би максимально відображав не тільки те, що хоче розробник системи, а й також враховував побажання користувачів. На знаходженні балансу між складність, функціональністю, простотою та зрозумілістю і ґрунтується сучасна теорія дизайну користувацьких інтерфейсів.

Процес розробки візуально привабливих та інтуїтивно зрозумілих інтерфейсів користувача для будь-якої системи чи застосунку відомий як дизайн інтерфейсу користувача або скорочено UI (user interfaces) користувача. Досвід користувача (user experience, UX), який описує всю взаємодію між користувачем та інтерфейсом, є ключовим компонентом UI.

Щоб створити інтуїтивно зрозумілий інтерфейс та подарувати користувачам безшовний досвід користування цим інтерфейсом, UI дизайнери повинні враховувати бажання та поведінку своїх користувачів. Під час процесу проєктування UI-дизайну створюються каркаси, прототипи та остаточні проєкти, включаючи візуальні компоненти, такі як колір, типографіка та макет.

У свою чергу процес створення користувацького досвіду продукту чи послуги відомий як UX-дизайн. Завдяки використанню інформаційної архітектури, графічного дизайну, тестування на зручність використання встановлюється плавна взаємодія між користувачем і продуктом.

Цілі UX-дизайну – покращити задоволення користувачів від користування системою, зацікавити їх і підвищити таким чином успіх компанії. Продукти, в яких використовується UX-дизайн, не тільки естетично привабливі та пропонують чудовий досвід користувача, але й прості в експлуатації.

Для того аби бути помітною в мережі Інтернет, системі просто необхідно мати правильний UI/UX-дизайн. Реклами та просування буде недостатньо, якщо

дизайн інтерфейсів буде відштовхувати новий користувачів, бути незручним, або мати відштовхуючу кольорову гаму. А для того аби спроєктувати такі інтерфейси, які будуть приваблювати користувачів існує наступний перелік порад:

– простота на першому місці;

– увага до зручності використання;

– швидкість має значення;

– оптимізована навігація;

– мобільна чутливість;

– пріоритет узгодженості;

– увага до доступності;

– фокус на потребах користувача;

– використання хорошої візуальної ієрархії;

– тестування та врахування відгуків користувачів.

Принцип «простота на першому місці» – найперша і найголовніша рекомендація для UI/UX-дизайну. Чудова взаємодія з користувачем пов’язана з простим дизайном. Система має бути простою для користувачів. Вони можуть сприймати надто складні та переповнені сторінки перевантаженими, особливо якщо вони не знайомі із системою. Спрощені та нескладні інтерфейси користувача набагато зручніші в користуванні, бо не дозволяють загубитися у функціоналі. Дизайн може вважатися простим у виконанні, коли в ньому послідовно використовуються шрифти та колірні схеми, чіткі лінії та ефективна візуальна ієрархія.

Принцип «увага до зручності використання» визначає прискіпливе ставлення до того, чи користувачеві зручно користуватися системою чи ні. Якщо ж йому не сподобається взаємодія, він перейде до іншої більш зручної системи, навіть якщо та буде мати менше функціоналу чи більшу вартість послуг. Для того, щоб гарантувати, що система є зручною для користувача, дизайнери повинні добре розуміти потреби цільової аудиторії. Наприклад, веб інтерфейси, призначені для старшої аудиторії, часто вимагають більших пунктирних шрифтів для кращої читабельності, тоді як інтерфейси, призначені для молодшої аудиторії,

часто можуть бути більш лаконічними та використовувати менші розміри шрифтів.

Наступний принцип «швидкість має значення» полягає у тому, щоб сторінки веб-інтерфейсу системи завантажувались швидко. Якщо сторінка буде довго завантажуватись, або виконання певних операцій буде затягуватись у часі, користувач може роздратуватися через довге очікування. Для появи дискомфортних відчуттів достатньо затримки всього у 10 секунд. Окрім дратування відвідувачів, повільне завантаження може знизити рейтинг сервісу в пошуковій системі. Тому слід дотримуватися рекомендованих практик для взаємодії з користувачем. Для цього необхідно переконатися, що система оптимізована за швидкістю завантаження та відповіді. Задля покращення швидкості взаємодії можна мінімізувати кількість HTTP-запитів, оптимізувати розміри зображень і позбутися величезних медіафайлів або зображень, які спричиняють повільне завантаження.

Принцип «оптимізації навігації» передбачає перевірку факту, що користувачі можуть просто орієнтуватися у системі сервісу. Для цього необхідно створити просту та легку навігаційну систему. У всьому UI/UX-дизайні системи структура навігації має бути послідовною та простою у використанні. Крім того, слід додати впізнавані індикатори, які спрямовують користувачів і дозволяють їм легко визначити, де вони знаходяться.

«Мобільна чутливість» – це, по суті, адаптованість дизайну інтерфейсів під мобільні пристрої. Оскільки переважним пристроєм для доступу до веб-сервісів є саме мобільні пристрої, вкрай важливо, щоб інтерфейси системи реагували на них та підлаштовувалася під них.

Немаловажливим аспектом користувацького досвіду є узгодженість та послідовність взаємодії з веб-сервісом. Споживачі хочуть, щоб їхній досвід був узгодженим на різних пристроях і сторінках. Це означає, що усі однакові елементи інтерфейсу повинні мати однакові функції та однакові патерни взаємодії з ними по всьому сервісу. Для забезпечення узгодженості найкраще

використовувати однакові дизайни та патерни поведінки для окремих видів елементів: кнопок, списків, форм тощо.

Також не слід забувати і про доступність, яка визначається наявністю спеціальних можливостей, і важливою, хоч і не першочерговою необхідністю, для UI/UX-дизайну. Для людей з вадами, які можуть зіткнутися з труднощами під час використання веб-сервісу, наявність подібних функцій є надзвичайно важливою. Потрібно переконатися, що система має всі необхідні елементи для полегшення навігації для людей з фізичними вадами, вадами зору або слуху. Важливо включити дизайн доступності на ранній стадії процесу, оскільки потім може бути складно його адаптувати постфактум.

Іншим важливим фактором є зосередження фокусу уваги на потребах користувачів. Для того, щоб визначити, чого люди хочуть від системи та які у них проблеми, потрібно провести дослідження користувацької бази. Отримавши ці знання, можна включити ці потреби в дизайн кінцевого продукту.

Ще однією практикою хорошого UI/UX-дизайну є впорядкування елементів у візуальну ієрархію відповідно до їх відносної важливості. Добре продумана візуальна ієрархія має важливе значення для того, щоб вести споживачів через систему і надавати їм візуальний контекст. Використання розміру, кольору та розташування для спрямування уваги користувача на найважливіші компоненти на сторінці є прикладом хорошої візуальної ієрархії. Наприклад, щоб кнопка із закликом до дії виділялася, вона має бути більшою та яскравішою за інші елементи на сторінці.

І останнім у списку, але не зазначенням, іде тестування та врахування відгуків користувачів. Коментарі користувачів можуть надати безцінну інформацію про те, чи успішним є дизайн сервісу. За допомогою використання A/B-тестування, опитування користувачів і прототипів дизайнери можуть визначити, чи дизайн задовольняє встановлені вимоги зручності та простоти використання. Для розробника важливо бути сприйнятливим до конструктивної критики та пропозицій від споживачів і відповідно змінювати дизайн системи.

Враховуючи усі вище зазначені поради, а також функціонал системи, що був розглянутий у 3.2 та 3.3, було спроєктовано і розроблено інтерфейси користувача.

Зайшовши на веб сайт системи, користувач в першу чергу зустрінеться з її початковою сторінкою (рис. 4.6). Ця сторінка міститиме форму авторизації, а також поля верхнього та нижнього колонтитулів (header та footer). Верхня панель – це панель навігації, за допомогою якої користувач може переміщатися між розділами «Карта», «Дрони» та «Політика конфіденційності». Окрім цього в ній міститься іконка профілю користувача, яка при натисканні видаватиме вибір налаштувань профіля, налаштувань планів підписки, а також опції входу або виходу з системи в залежності від того, чи користувач вже авторизований чи ні.

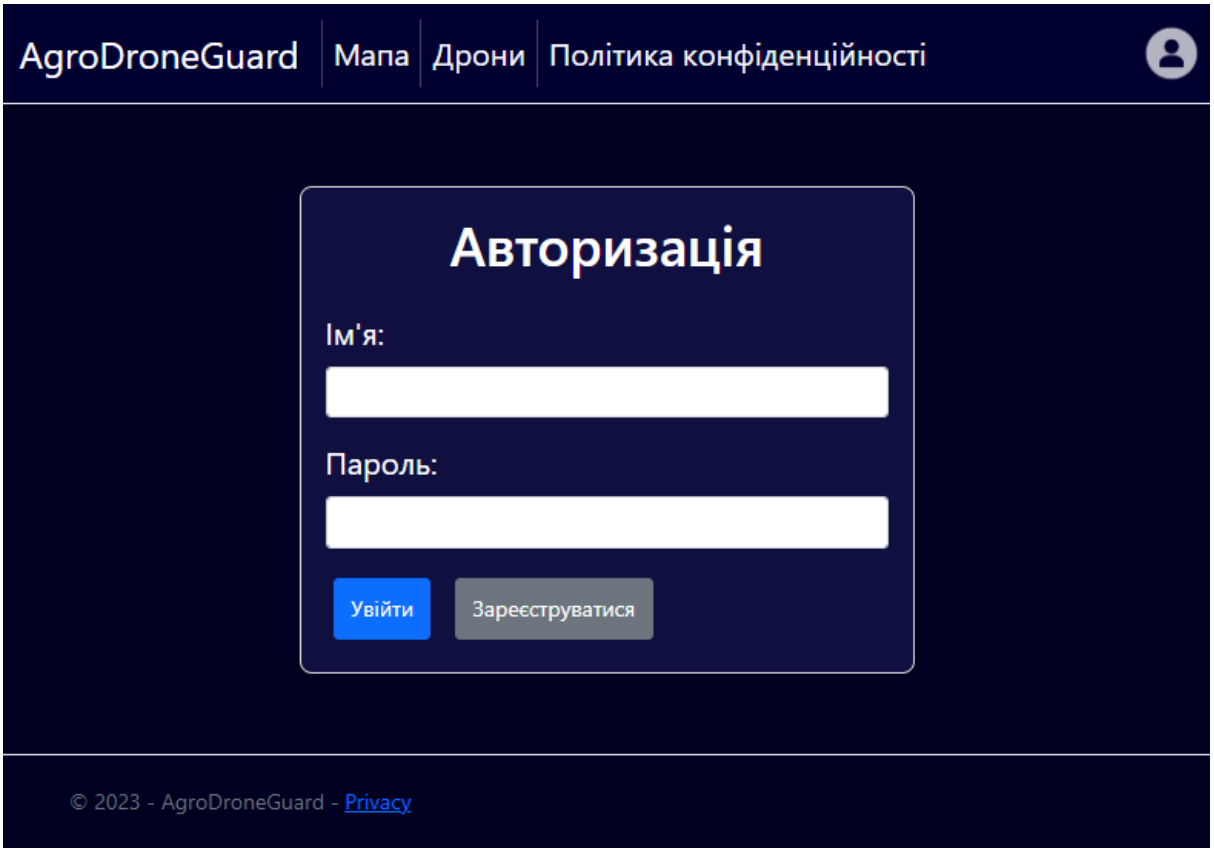

Рисунок 4.6 – Інтерфейс авторизації в системі

Якщо ж користувач ще не зареєстрований у системі, він може натиснути на кнопку «Зареєструватися», після чого він потрапить на сторінку з формою реєстрації (рис. 4.7). У цій формі користувач може вказати ім’я (яке буде

використано як логін) та пароль, а також йому буде необхідно підтвердити пароль, щоб запобігти випадковій помилці під час вводу.

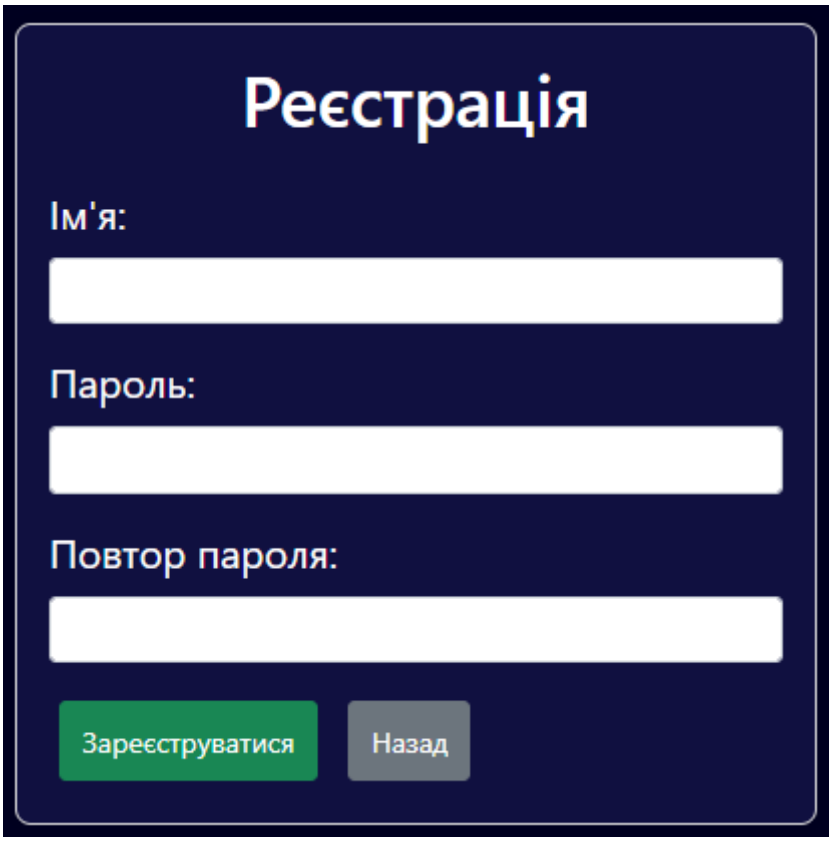

Рисунок 4.7 – Інтерфейс реєстрації в системі

Після успішного входу в систему користувачу відкривається сторінка роботи з картою (рис. 4.8). Сторінка поділена на дві частини: дві третини займає власне карта, а ще одна третина відведена під панель керування. Карта використовує зображення з Bing Maps для відображення вигляду із супутника. Під час завантаження сторінки на карту також підвантажуються усі об’єкти (РО та ТЗ), і після завантаження клієнт з періодичністю у 5 секунд надсилає запит на сервер для оновлення цього списку об’єктів.

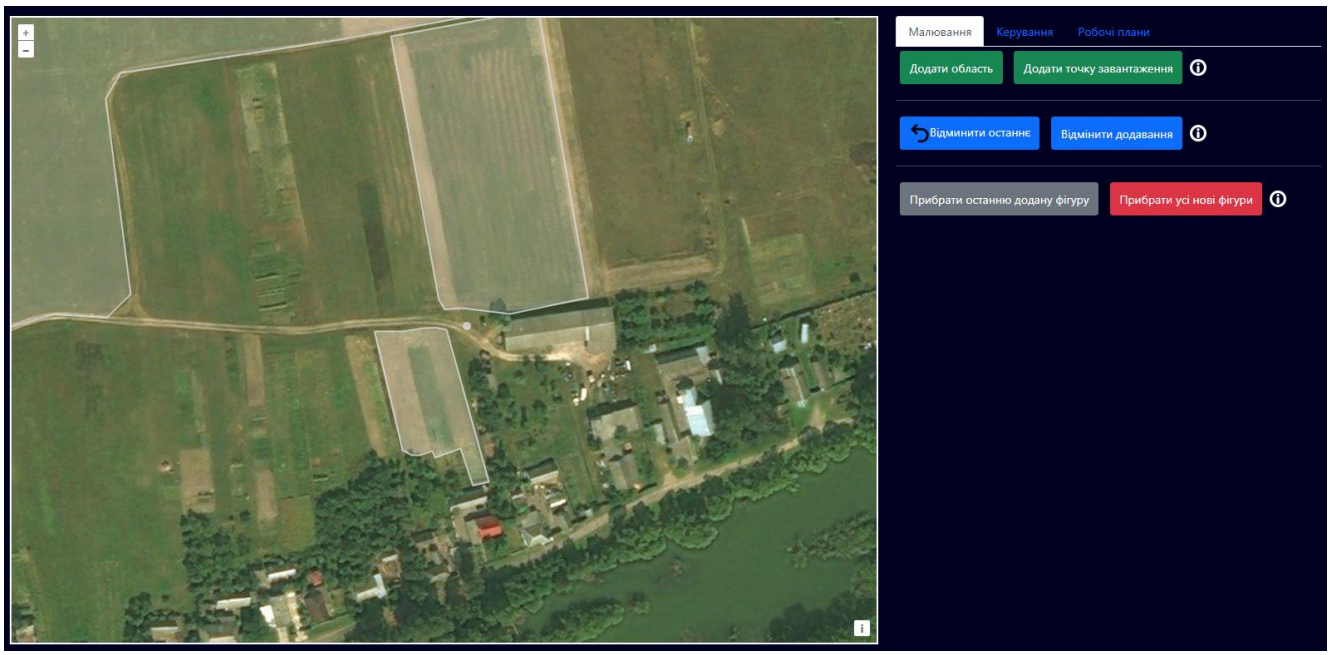

Рисунок 4.8 – Інтерфейс роботи з картою. Вкладка «Малювання»

На панелі керування за замовчуванням відкривається вкладка «Малювання». У цій вкладці користувач може обрати функції додавання РО та ТЗ, відміни останньої дії (якщо це стосується нанесеної точки під час формування полігону РО) або ж відміни додавання (у разі якщо користувач передумав додавати об’єкт, і хоче вийти з режиму додавання). Також тут присутні кнопки для видалення останньої доданої фігури та видалення усіх новододаних фігур. Окрім цього інтерфейс має зручні довідкові пояснення для кожної кнопки, які відображаються у разі наведення вказівника на відповідну іконку зі знаком оклику (рис. 4.9).

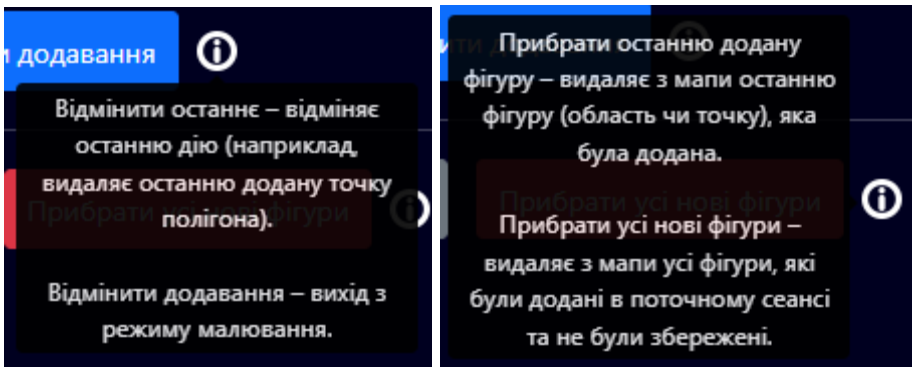

Рисунок 4.9 – Довідка до функціоналу

Наступною вкладкою є вкладка «Керування». Вона відповідає за керування об’єктами на карті. За замовчуванням в ній міститься лише одна кнопка «Обрати фігуру» (рис. 4.10). Натиснувши на неї, користувач зможе виділяти об’єкти на карті, як це продемонстровано на рисунках 4.11 та 4.12. При наведенні мишкою на фігуру (рис. 4.11), вона змінить своє відображення задля індекації можливості її вибору, а при натисканні на таку фігуру лівою кнопкою миші вона буде обрана (рис. 4.12). Після обрання або РО, або ТЗ, або одночасно двох фігур цих типів, в панелі керування відкриється форма збереження виділених об’єктів в системі (рис. 4.13). Відповідно до того, чи буде вибрано одну лише РО або ТЗ, чи обидві фігури разом, будуть відображатися й поля у формі.

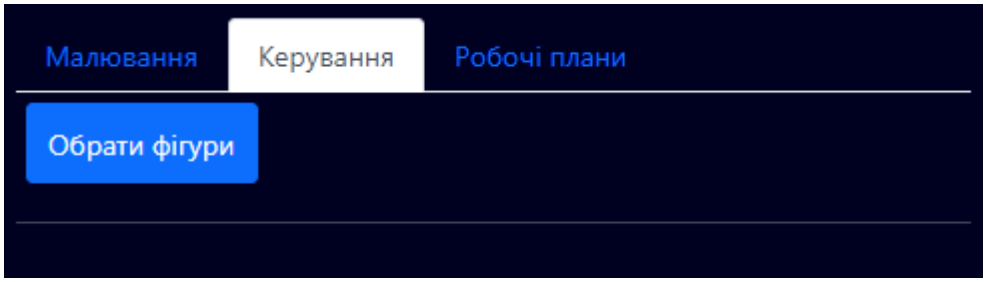

Рисунок 4.10 – Вигляд вкладки «Керування» за замовчуванням

Рисунок 4.11 – Зміна вигляду фігур при наведенні на них мишкою під час дії режиму вибору фігур

Рисунок 4.12 – Зміна вигляду фігур при їх обранні

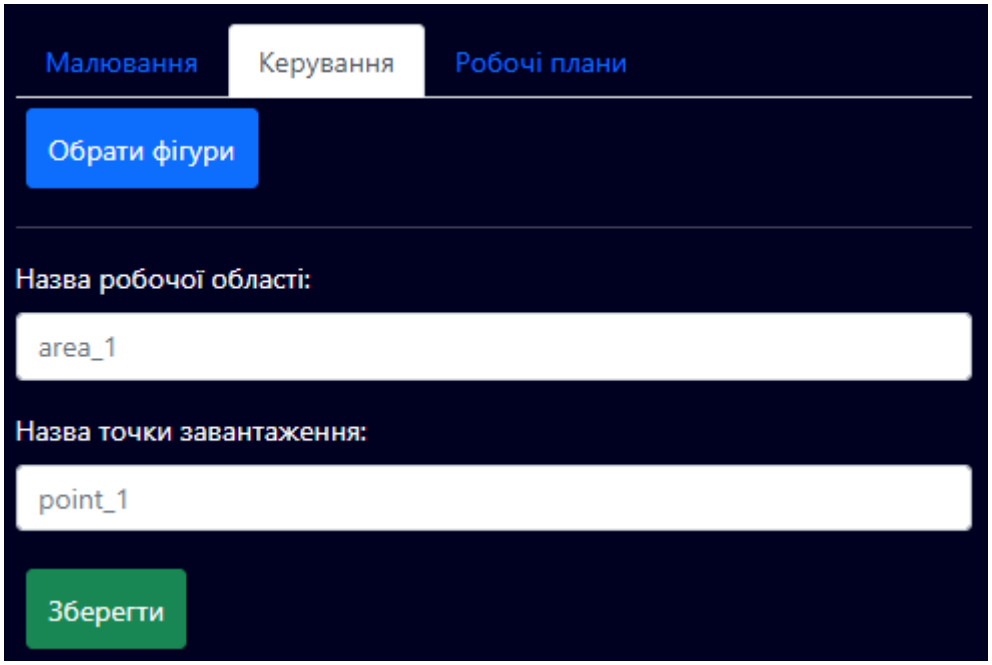

Рисунок 4.13 – Форма для збереження виділених об’єктів

Окрім цього, якщо обрати лише один об’єкт, то на панелі керування з’явиться ще одна кнопка «Видалити виділене» (рис. 4.14). По її натисканню інформація про виділений об’єкт буде надіслана на сервер, і якщо такий об’єкт присутній в системі, його буде видалено, інакше ж буде видано помилку.

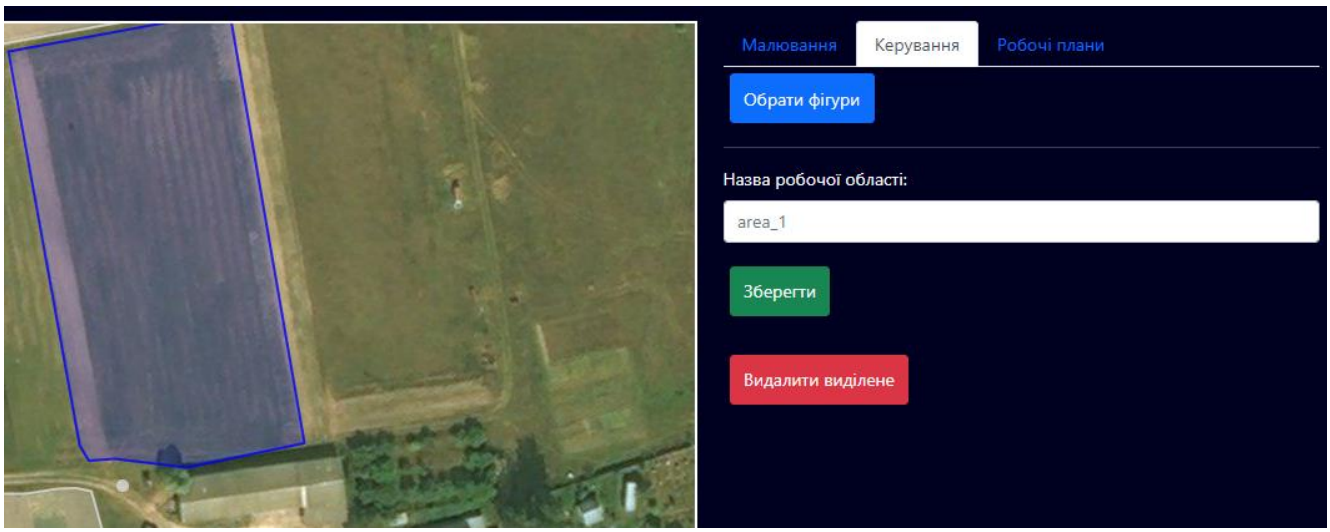

Рисунок 4.14 – Можливість видалення об’єкту з системи

Останньою в панелі керування на цій сторінці є вкладка «Робочі плани» (рис. 4.15). На вкладці присутні усі необхідні функції. Тут можна перейти до форми створення нового РП, натиснувши на кнопку «Створити план». Можна також вибрати план зі списку існуючих, та запустити, редагувати або видалити його, натискаючи на відповідні кнопки.

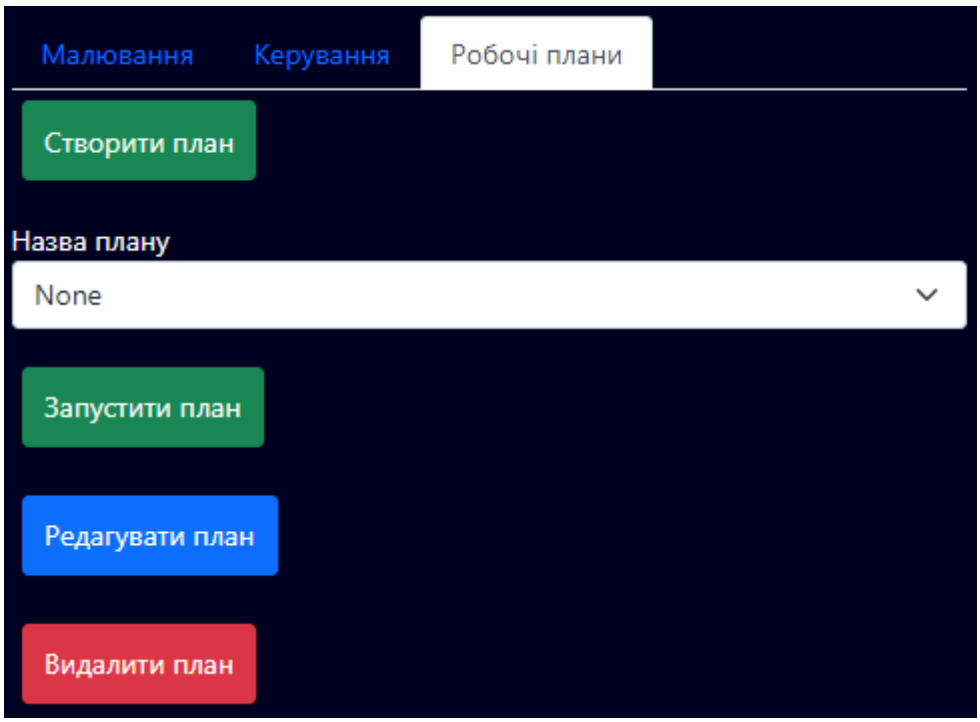

Рисунок 4.15 – Вигляд вкладки «Робочі плани»

Якщо перейти до створення плану, можна буде побачити форму, яка зображена на рисунку 4.16. За замовчуванням вона має поля для назви плану, РО та ТЗ, а також список з дронами. Поля з назвами РО та ТЗ доступні тільки для читання, оскільки заповняються вони через виділення необхідних об’єктів на карті через відповідні кнопки. Також після вибору дрона зі списку у формі відкриваються додаткові параметри для налаштування (рис. 4.17), які напряму пов’язані з моделлю обраного дрона.

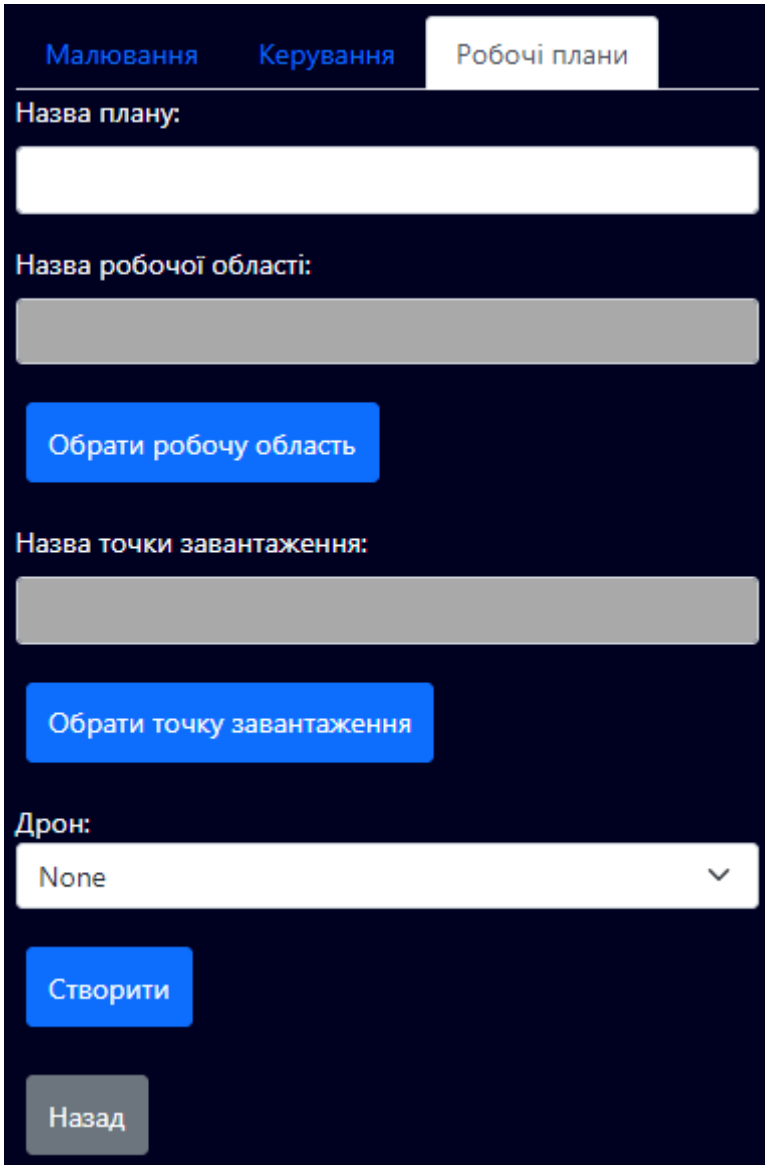

Рисунок 4.16 – Форма створення нового плану

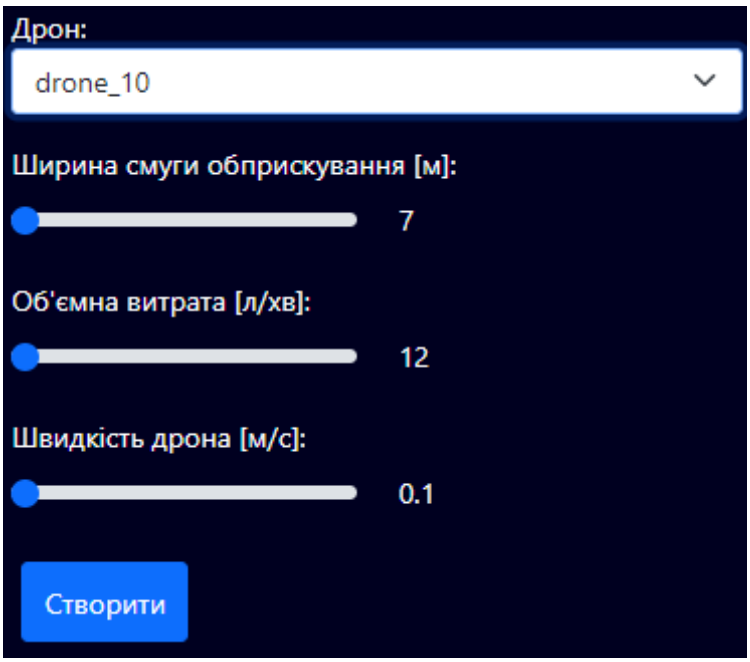

Рисунок 4.17 – Налаштування параметрів плану

Після створення плану для нього буде створено ТОП відповідно до тих РО, ТЗ та параметрів, які були задані. Після цього його можна обрати зі списку, і одразу побачити ТОП, яка була побудована (рис. 4.18).

Після обрання плану можна перейти до його редагування. Форма редагування майже не відрізняється від форми створення за виключенням наявності додаткового поля з поточною назвою плану (рис. 4.19).

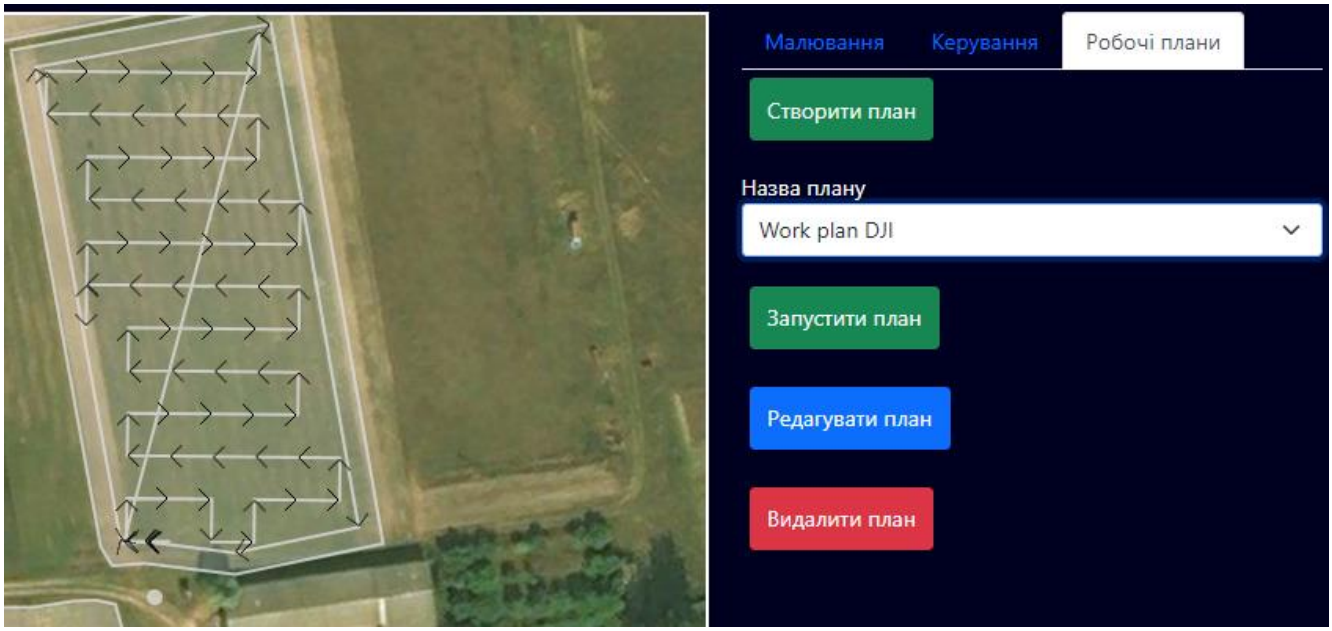

Рисунок 4.18 – Відображення ТОП для вибраного плану

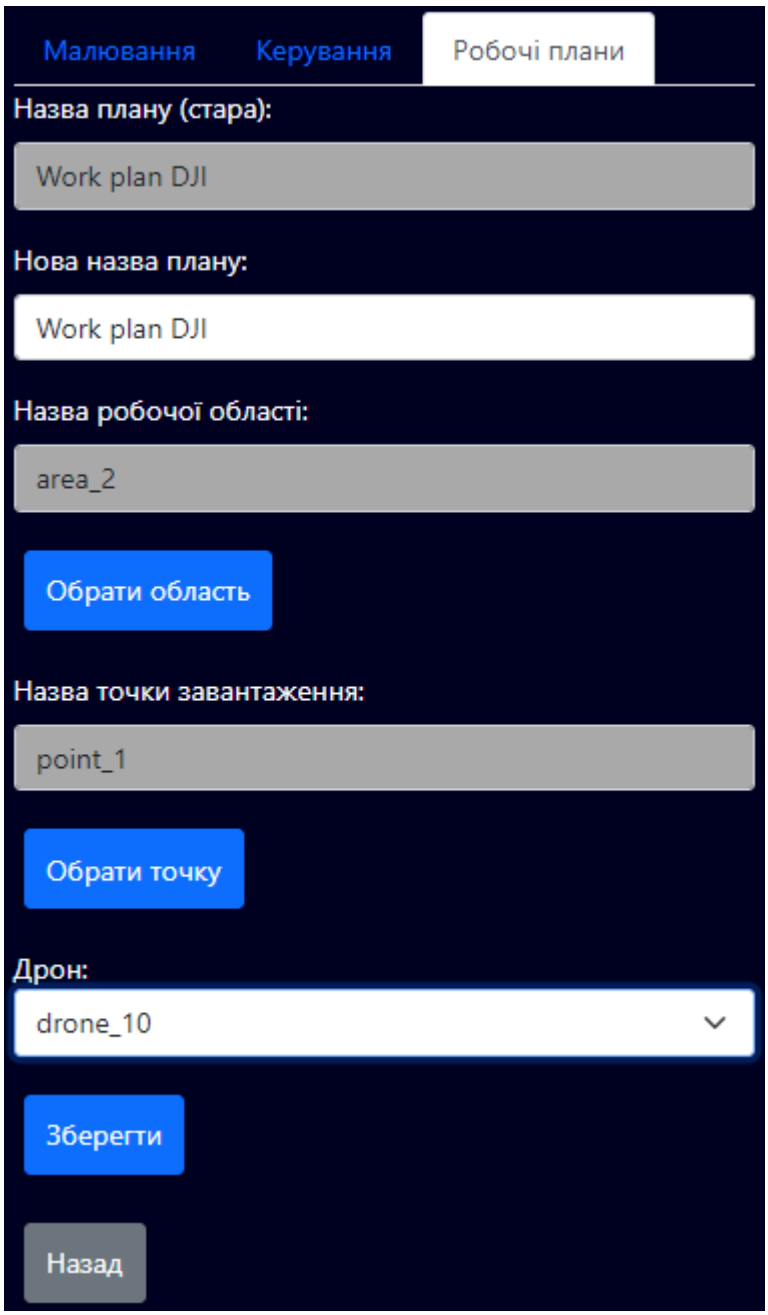

Рисунок 4.19 – Форма для редагування РП

А по натисканню на кнопку «Запустити план» обраний план буде запущено в дію. Після цього на карті можна побачити, як з ТЗ відправиться дрон і літатиме за ТОП. Одночасно можна запускати не один план, а одразу декілька, які будуть виконуватися паралельно (рис. 4.20).

Окрім сторінки роботи з картою в системі також присутня сторінка керування дронами. Вона має 4 вкладки, першою з яких є вкладка керування моделями дронів (рис. 4.21).

Рисунок 4.20 – Процес виконання двох планів одночасно

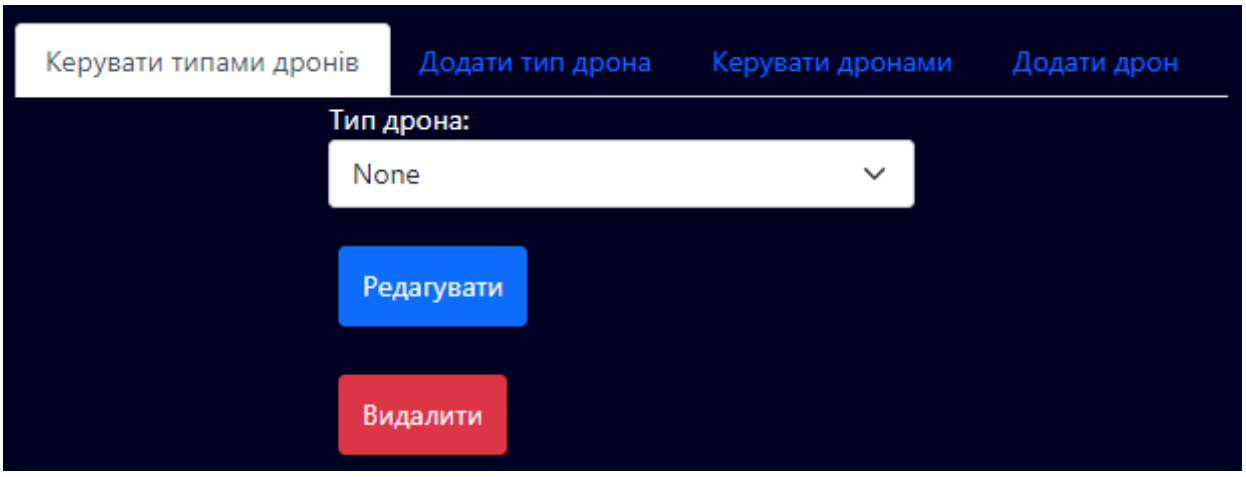

Рисунок 4.21 – Вигляд вкладки «Керувати типами дронів»

У разі вибору моделі дрона зі списку, у правій частині з’являється таблиця з описом вибраної моделі (рис. 4.22). Також у цій вкладці можна редагувати інформацію обраної моделі, або ж взагалі видалити її з системи.

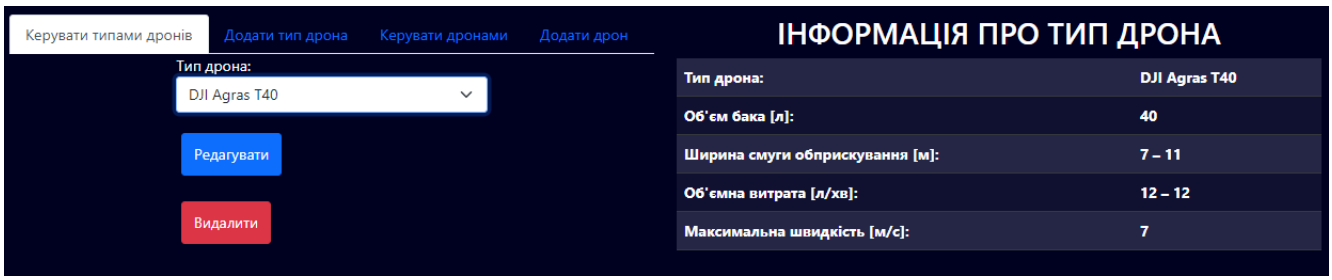

Рисунок 4.22 – Інформація про модель дрона

Наступною вкладкою є вкладка «Додати тип дрона», за допомогою якої можна додати нову модель дрона до системи (рис. 4.23). Форма має поля назви моделі дрона, об’єм його резервуара, максимально та мінімально можливі показники ширини смуги обприскування та об’ємної витрати, а також максимальну швидкість дрона.

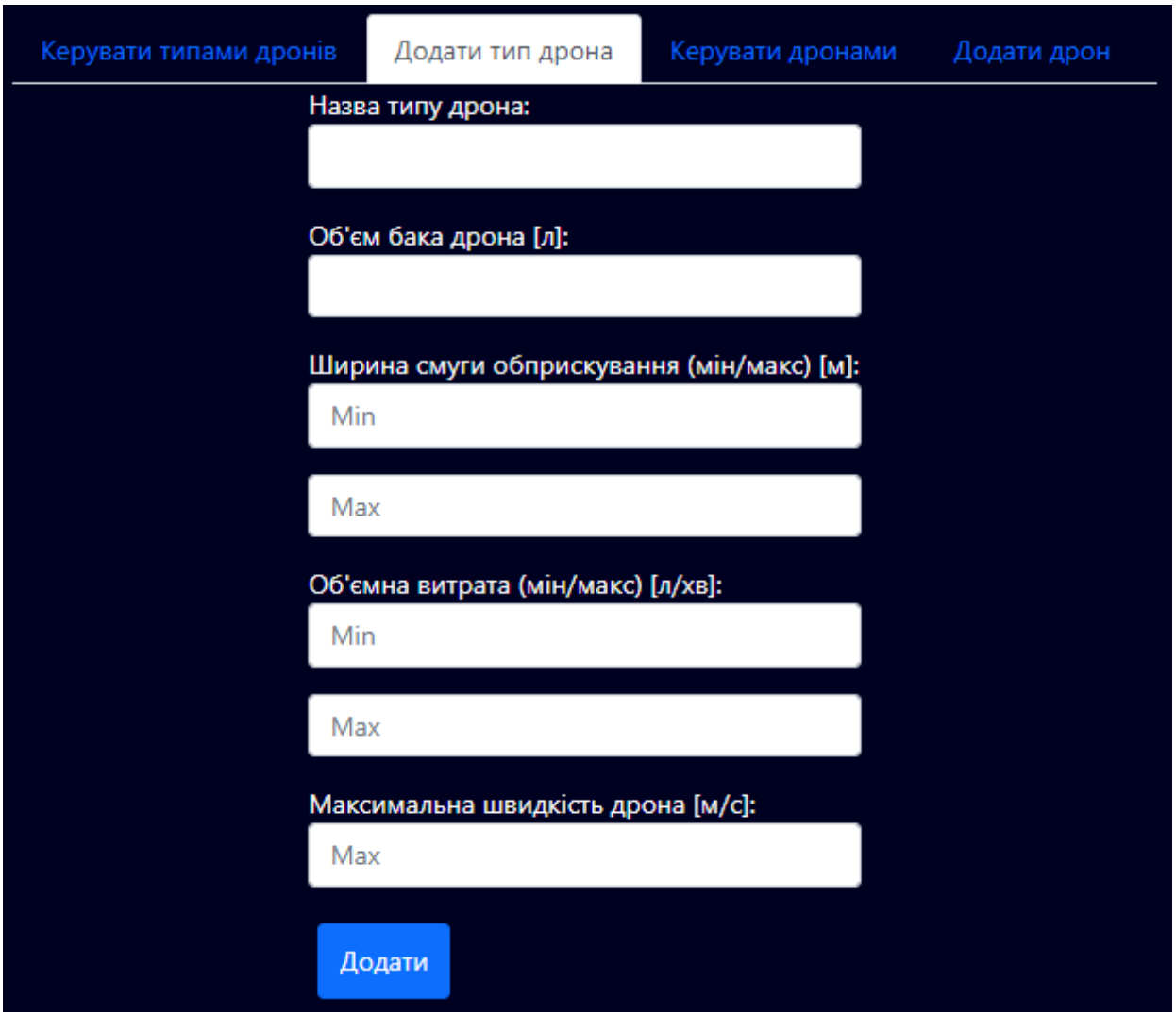

Рисунок 4.23 – Форма додавання моделі дрона

Вкладка «Керувати дронами» ідентична вкладці керування моделями. Вона так само має список для вибору дронів, кнопки для переходу до форми редагування та для видалення дрона. І так само при виборі дрона відображається таблиця з інформацією про його модель.

Останньою вкладкою є вкладка додавання нових дронів (рис. 4.24). Вона має поле для введення назви дрона та список для вибору його моделі. Тут так само при виборі моделі зі списку інформація про модель дрона відображається у таблиці справа.

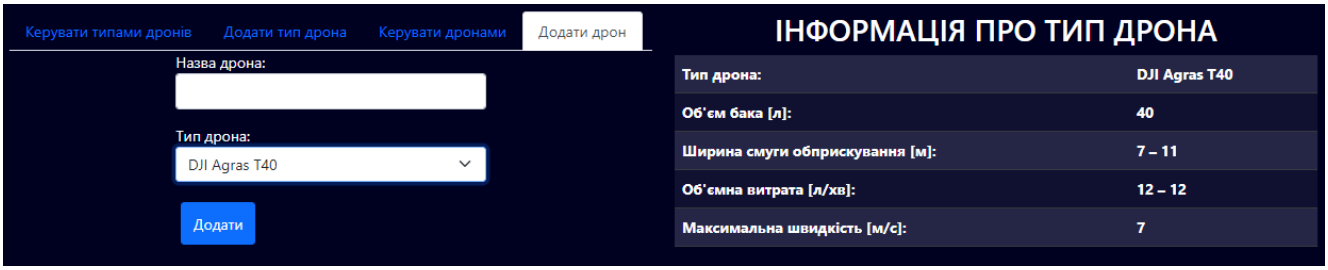

Рисунок 4.24 – Вигляд вкладки «Додати дрон»

Висновки до розділу 4

У цьому розділі описано загальну структуру системи, а також структуру та функціонал її компонентів, серед яких: API контролери, представлення, моделі, сервіси та база даних. Серед конролерів присутні контролер карти (відповідає за керування процесом взаємодії з картою, її об’єктами та робочими планами), контролер дронів (відповідальний за керування операціями додавання, видалення та редагування дронів та їх моделей) та контролер авторизації (відповідальний за процес авторизації та реєстрації в системі). З сервісів було реалізовано сервіс хешування паролів, керування дронами та роботи з БД. Також в цьому розділі розглянуто розроблену структуру БД. Окрім цього в ньому розглянуто розроблений алгоритм побудови траєкторії обходу покриття.

Заключною частиною розділу є розробка інтерфейсів користувача, в якому надано приклади розроблених інтерфейсів та опис їхнього функціоналу та процесу роботи з ними.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Ідеєю стартап-проєкту є створення інформаційної системи з підтримки обприскування сільськогосподарських полів за допомогою БПЛА задля того, аби надати аграріям України та світу продукт, який дозволить автоматизувати процес обприскування та заощадити час і ресурси. Детальний опис ідеї можна побачити у таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проєту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Інформаційна система, яка буде надавати користувачам можливість використовувати для обприскування полів більшість моделей дронів, що доступні на ринку. Система використовуватиме алгоритми для знаходження оптимальної траєкорії обходу покриття полів із залученням мінімальної кількості речовини та за мінімальний час.	Застосування для обприскування полів у сільськогосподарських підприємствах.	Зменшення часу, фінансових та ресурсих витрат на обприскування. Збільшення ефективності процесу обприскування. Зручніше та простіше обслуговування засобів обприскування.
	Застосування для обприскування полів для приватних осіб.	Можливість економії фінансів за допомогою використання недорогих моделей дронів, або дронів власного виробництва.

Ключовими відмінностями системи від вже існуючих на ринку рішень є можливість використання більшої кількості моделей дронів різних виробників (у

тому числі й дронів власного виробництва), покращені алгоритми побудови оптимальної траєкторії обходу покриття та більша ефективність у роботі з великими обсягами геотопологічних даних.

У таблиці 5.2 можна побачити порівняльну характеристику ідеї проєкту з вже існуючими рішеннями від потенційних конкурентів.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	Товари та концепції потенційних конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Hylio AgroSol			
1.	Алгоритми оптимізації траєкторії обходу покриття	Присутнє використання алгоритмів знаходження оптимальної траєкторії	Застосування лише методу бустрофедона			+
2.	Відсоток покриття ділянки	Дорівнює приблизно 99%	Точний відсоток невідомо, але не більше 95%			+
3.	Час відгуку на дії	Майже моментальний, але у деяких випадках операції можуть	Моментальний	+		

№ п/п	Техніко-економічні характеристики ідеї	Товари та концепції потенційних конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Hylio AgroSol			
		зайняти певний час				
4.	Зручність використання	Зручний та мінімалістичний інтерфейс зі довідковими та інтерактивними підказками.	Лаконічний та послідовний інтерфейс		+	
5.	Доступність	Доступний з браузера на будь-якому пристрої з підключенням до мережі Інтернет	Автономний застосунок для ОС Windows			+
6.	Вартість послуг	Помірковані ціни з урахуванням середньої заробітної плати в Україні	Середні ціни для регіону США			+

5.2 Технологічний аудит ідеї проєкту

Аби реалізувати ідею проєкту необхідним кроком є проведення аудиту технологій, які можуть бути використані для реалізації цього проєкту. Першим пунктом в аудиті стоїть визначення того, чи проєкт взагалі можливо реалізувати, використовуючи необхідні для нього технології. Для цього необхідно визначити, чи вони вже існують, чи їх потрібно розробляти з нуля або дороблювати, а також треба визначити, чи є ці технології доступними. Відповіді ні ці питання продемонстровані у таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Інформаційна система, яка буде надавати користувачам можливість використовувати для обприскування полів більшість моделей дронів, що доступні на ринку. Система використовуватиме алгоритми для знаходження оптимальної траєкорії обходу покриття полів із залученням мінімальної кількості речовини та за мінімальний час.	Фреймворк ASP.NET Core MVC	Наявна	Доступна
2.		Карти OpenLayers	Наявна	Доступна
3.		Пакувальний Webpack	Наявна	Доступна
4.		База даних DynamoDB	Наявна	Доступна

З результатів аудиту, що зображені у таблиці, можна побачити, що усі ключові технології, необхідні для реалізації ідеї, є наявними на ринку та доступними для авторів проєкту.

З технологій було обрано фреймворк ASP.NET Core MVC, карти OpenLayers, пакувальник Webpack та базу даних DynamoDB через їх переваги, такі як мультиплатформність, безкоштовність, та швидкість роботи з великими обсягами запитів та даних.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Оскільки проєкт, який розробляється авторами, може бути не єдиним, який надає схожі послуги на ринку, необхідно провести аналіз ринкових можливостей та загроз, що можуть виникнути під час запуску стартап-проєкту.

Першим кроком проводиться аналіз попиту. Результати цього аналізу представлені у таблиці 5.4.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	1
2.	Загальний обсяг продаж, грн/ум.од	-
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	-
5.	Специфічні вимоги до стандартизації та сертифікації	Необхідність отримання дозволу на політ дронів від місцевих органів правопорядку під час воєнного стану

6.	Середня норма рентабельності в галузі (або по ринку), %	20

Виходячи з результатів аналізу попиту, можна зробити висновок, що ринок є привабливим для входження.

Наступним кроком є визначення потенційних груп клієнтів, їх характеристики, а також орієнтовного переліку вимог до сервісу для кожної групи. Отриману характеристику можна побачити у таблиці 5.5.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	На ринку є потреба для системи, яка б дозволяла здійснювати обприскування дронами автономно (без залучення оператора), та до якої був би доступ з більшості портативних пристроїв	Малі, середні та великі сільськогосподарські підприємства	Різниця у розмірах дронів, і як результат у фінансах, які витрачаються на них	Швидкодія роботи системи
2.				Стабільність та коректність роботи системи
3.				Функціональність
4.				Можливість використання різних моделей дронів від різних виробників

Третім кроком йде аналіз ринкового середовища. До цього аналізу входять визначення факторів загроз (табл. 5.6) та факторів можливостей (табл. 5.7), які тим чи іншим чином негативно або позитивно впливають на впровадження проєкту на ринку.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Підвищення конкуренції	Поява додаткових гравців на ринку, які надаватимуть схожі або ідентичні послуги	Додавання нового функціоналу. Оновлення дизайну для більшого залучення клієнтів. Перегляд цінової політики підписок
2.	Зміна потреб клієнтів	Клієнтам необхідні додаткові функції звітності, аналізу, збереження інформації про всі обприскування у довгостроковій перспективі	Додавання нового функціоналу
3.	Воєнні дії	Загроза для головного офісу компанії та її співробітників у зв’язку з воєнними діями	Релокація обладнання та персоналу до завчасно підготовленого додаткового офісу в безпечній зоні

Наступним кроком йде аналіз пропозиції, в якому визначаються загальні риси конкуренції на ринку. Результати цього аналізу можна побачити у таблиці 5.8.

Після цього також зазвичай проводиться детальний аналіз умов конкуренції в галузі за моделлю 5 сил М. Портера. Його результати можна переглянути у таблиці 5.9.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Відсутність конкуренції	Майже повна відсутність аналогів, або відсутність аналогів з необхідним функціоналом чи локалізацією	Проведення рекламних заходів. Залучення нових клієнтів.
2.	Сплеск розвитку сільськогосподарської діяльності	Велике та стрімке зростання кількості сільсько-господарських підприємств та (або) модернізації сільсько-господарської діяльності	Проведення рекламних заходів. Активний пошук нових клієнтів

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції – чиста	На ринку присутня певна кількість гравців, які надають однакові за своєю суттю послуги, але їхні розміри не покривають усього попиту	Можливість легкого виходу на ринок та легкого завоювання власної ніші на ньому
Рівень конкурентної боротьби – міжнародний	Компанії-конкуренти здебільшого перебувають у різних країнах світу	Локалізація продукту найбільш популярними мовами. Проведення рекламних кампаній в інших країнах

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
За галузевою ознакою – внутрішньогалузева	Конкуренція відбувається у рамках галузі сільського господарства	Орієнтація проєкту на аграрії
За видом товарів – товарно-видова	Конкуренція між сервісами, що надають схожі послуги	Створення продукту, який враховуватиме недоліки та переваги конкурентів та матиме ключові переваги над ними
За характером конкурентних переваг – нецінова	Клієнтам важлива не стільки ціна, скільки функціонал сервісу	Додавання нового раніше недоступного функціоналу та покращення вже існуючого
За інтенсивністю – не марочна	Як такого відомого бренду на ринку не існує	Збільшення уваги до продукту за рахунок реклами та участі у різноманітних виставках та конференціях

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Hylio AgroSol	Обмеження застосування БПЛА під час дії воєнного стану.	Постачання дронів з країн виробників	Клієнти визначають основний попит на продукцію. У разі незадоволення якістю сервісу, готові перейти до інших продуктів, або відмовитися від застосування дронів взагалі	Ширший функціонал з боку товарів-замінників
Висновки	Інтенсивність конкуренції невисока у зв’язку з невеликою кількістю конкурентів	Присутня можливість виходу на ринок. Потенційних конкурентів немає. Термін виходу на ринок – 12	Постачальники дронів диктують умови їх розповсюдженості на території України та можуть неочікувано перервати поставки у	Клієнти диктують умови функціональності та зручності сервісу та його ціни	Обмеження майже відсутні, оскільки товарів-замінників невелика кількість

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
		місяців.	зв’язку з ускладненням логістики або через політичні причини		

З огляду на результати аналізу конкуренції, продемонстрованого у таблицях 5.8 та 5.9, можна зробити висновок, що вихід на ринок досить вірогідно буде відносно легким. Для того, щоб це справдилося, сервіс повинен обходити своїх прямих та непрямих конкурентів у функціоналі, який він надає, та мати зрозумілий та зручний інтерфейс, локалізований більшістю найпопулярніших мов.

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проєкту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1.	Ціна	Помірна цінова політика сприятиме залученню нових клієнтів
2.	Функціональність	Чим більше функцій виконує система, тим ймовірніше виберуть саме її
3.	Доступність	Якщо сервіс доступний на багатьох платформах, ним буде значно зручніше користуватися, і як результат, він матиме більше охоплення та попит

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
4.	Використання різних моделей дронів від різних виробників	Надає сервісу перевагу над конкурентами, оскільки дозволяє клієнтам не адаптуватися під вимоги системи, і використовувати найбільш вигідні для них моделі дронів, або вже ті, які в них є в наявності

Далі відповідно до визначених факторів конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проєкту. Його можна побачити у таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін інформаційної системи з підтримки обприскування сільськогосподарських полів за допомогою БПЛА

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з інформаційною системою з підтримки обприскування сільськогосподарських полів за допомогою БПЛА						
			-3	-2	-1	0	+1	+2	+3
1.	Ціна	15			+				
2.	Функціональність	12				+			
3.	Доступність	18		+					
4.	Використання різних моделей дронів від різних виробників	10				+			

Фінальним етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 5.12) на основі

виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 5.11).

Таблиця 5.12 – SWOT- аналіз стартап-проєкту

Сильні сторони: Ціна Доступність	Слабкі сторони: Функціональність Можливість використання різних моделей дронів від різних виробників
Можливості: Відсутність конкуренції Сплеск розвитку сільськогосподарської діяльності	Загрози: Підвищення конкуренції Зміна потреб клієнтів Воєнні дії

Як можна побачити з таблиці 5.11, стартап-проєкт має як і переваги (ціна, доступність), так і слабкі сторони (функціональність та можливість використання різних моделей дронів від різних виробників). Слабкі сторони не обов'язково визначають те, що він є чимось гірший за товари-конкуренти, але якщо він і не є кращим за них, то це вже можна вважати слабкістю. Тому необхідним є дороблення цих аспектів до прийнятного стану, після чого можна буде розмірковувати про випуск продукту на ринок.

Також слід враховувати можливості та загрози, які можуть виникнути під час запуску чи роботи проєкту, і завчасно підготувати перелік дій, які будуть виконані у разі настання якоїсь з цих подій.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проєкту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи можна побачити у таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Випуск сервісу з обмеженим функціоналом, з подальшим його вдосконаленням	60%	6 місяців
2.	Випуск сервісу одразу з повним функціоналом	80%	12 місяців

Серед двох альтернатив найкращим варіантом буде обрати першу. Якщо занадто зволікати з випуском проєкту на ринок, можна проґавити появу нових конкурентів, або пропустити сільськогосподарський сезон та період обприскувань. Якщо ж продукт випустити з мінімально прийнятним функціоналом, він хоч і залучить потенційно менше клієнтів, але таким чином у компанії одразу з’являться додаткові ресурси на посилення та пришвидшення розробки іншого функціоналу.

5.4 Розроблення ринкової стратегії проєкту

Розроблення ринкової стратегії є невід’ємним етапом запуску будь-якого стартап-проєкту. Першим кроком воно передбачає визначення стратегії охоплення ринку, а саме опис цільових груп потенційних споживачів. Цей опис можна побачити у таблиці 5.14

На основі огляду цільових груп основними, на яких слід зосередити увагу, було вибрано середні та малі сільськогосподарські підприємства. По-перше, хоч вони і не мають великих бюджетів, і не можуть собі дозволити велику кількість та велику ціну дронів, вони не настільки залежні від вже усталеної системи обробки полів та наявної техніки, і є більш гнучкими у цьому питанні. А по-друге, в цей сегмент легше зайти, та він не має такої сильної конкуренції.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Великі сільськогосподарські компанії	Середня	Середній	Висока	Складно
2.	Середні за розміром сільськогосподарські компанії	Висока	Високий	Середня	Відносно просто
3.	Малі сільськогосподарські компанії	Висока	Високий	Середня	Відносно просто
4.	Приватні особи	Низька	Низький	Низька	Складно

Оскільки обрано не одну цільову групу, за стратегію охоплення ринку обрано стратегію диференційованого маркетингу.

Для подальшої роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. Її визначення можна побачити у таблиці 5.15. Базовою стратегією розвитку було обрано стратегію диференціації, оскільки головною рисою проєкту є покращений функціонал та пропонування кращого користувацького досвіду.

Після цього йде вибір стратегії конкурентної поведінки. Результат цього вибору можна побачити у таблиці 5.16. За стратегію конкурентної поведінки було взято стратегію наслідування лідера, оскільки головним пріоритетом на першому етапі є зайняття ринку в Україні, де закордонні конкуренти майже не представлені.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Випуск сервісу з обмеженим функціоналом, з подальшим його вдосконаленням	Просування сервісу в пошукових системах. Пропонування сервісу напряму компаніям.	Цінова політика, доступність сервісу.	Стратегія диференціації

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні, є як мінімум один аналог на міжнародному ринку	Компанія шукатиме нових користувачів	Компанія копіюватиме загальні функції конкурентів, такі як: створення РО та ТЗ, побудова ТОП, додавання дронів, створення та запуск РП.	Стратегія наслідування лідера

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проєкт. Результати зображені у таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1.	Швидкодія роботи системи	Стратегія диференціації	Висока швидкість відгуку системи	Швидкість. Функціональність. Простота.
2.	Стабільність та коректність роботи системи		Висока стабільність роботи за будь-яких обставин	
3.	Функціональність		Більша кількість функцій	
4.	Можливість використання різних моделей дронів від різних виробників		Надання можливості	
5.	Зручний та зрозумілий інтерфейс		Лаконічний та інтуїтивний інтерфейс	

Результатом виконання підрозділу є узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

5.5 Розроблення маркетингової програми стартап-проєкту

Для того аби продажі продукту були задовільними, потрібно мати попит на свій товар. А для того аби стимулювати попит, та для підвищення впізнаваності бренду необхідно рекламувати свій продукт. Досягненням цієї мети і займається маркетинг. Тому дуже важливо мати продуману маркетингову програму для виходу на ринок, щоб мати якнайбільшу та якнайшвидшу окупність цього кроку.

Першим кроком у побудові маркетингової програми є формування маркетингової концепції товару, який отримає споживач. Так у таблиці 5.18 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Швидкодія	Висока швидкодія	Забезпечується за рахунок використання DynamoDB
2.	Функціональність	Висока функціональність	Більша кількість функцій у порівнянні з прямими конкурентами
3.	Можливість використання різних моделей дронів від різних	Можливість присутня	Можливість присутня на відміну від товарів-конкурентів

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
	виробників		
4.	Зручний та зрозумілий інтерфейс	Інтерфейс відповідає вимогам зручності, зрозумілості та простоти інтерфейсу	Використання новітніх методів та підходів до побудови дизайну

Наступним кроком розробляється трирівнева маркетингова модель товару: уточнюється ідея послуги, її функціональні складові, особливості процесу її надання. Сформовану модель можна побачити у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Автоматизоване обприскування сільськогосподарських полів за допомогою дронів з функцією побудови оптимальної траєкторії та високою швидкодією		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Функція побудови оптимальної ТОП.	Нм	Тх
	2. Використання DynamoDB для забезпечення швидкодії	Нм	Тх
	3. Лаконічний та інтуїтивний дизайн	Нм	Тх
	Якість: Надійний захист персональних даних та паролів. Висока швидкодія. Простота використання.		
	Пакування: Відсутнє (не є фізичним продуктом)		

Рівні товару	Сутність та складові
	Марка: «AgroDroneGuard» – інформаційна система з підтримки обприскування сільськогосподарських полів за допомогою БПЛА
ІІІ. Товар із підкріпленням	До продажу: відсутнє
	Після продажу: персональна підтримка з віддаленим або місцевим вирішенням проблем
За рахунок чого потенційний товар буде захищено від копіювання: за рахунок закритого вихідного коду системи та централізованої архітектури з вільним доступом лише до клієнтської частини.	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проєкту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 5.20). Аналіз проводиться експертним методом. Остільки більшість сервісів працюють за підписочною моделлю розповсюдження, усі ціни вказані за один платний період, який дорівнює одному місяцю. Рівень доходів споживачів вирахуваний як середньо статистичне значення чистого прибутку (включає вирахування усіх витрат) обраховане по усіх фермерських господарствах, які зареєстровані в Україні.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1.	7500 грн.	11100 грн.	170000 грн.	7000 – 10000 грн.

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 5.21): проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту); вибір та

обґрунтування оптимальної глибини каналу збуту; вибір та обґрунтування виду посередників.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Клієнти зацікавлені купівлею доступу до сервісу лише 1-2 рази на рік на обприскувальний сезон. Клієнти можуть бути зацікавлені у сервісі за наявності дронів у їхній власності або за можливості їхньої оренди в компанії.	Встановлення контактів із споживачами і підтримання їх. Проведення передпродажної підготовки, а також функції інформаційного й правового забезпечення.	Канал нульового рівня	Збут послуг через систему підписки з автоматичним стягуванням оплати щомісячно через мережу Інтернет.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів. Результати представлені у таблиці 5.22.

Результатом цього пункту є ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проєкт, та відповідну обрану альтернативу ринкової поведінки.

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Потенційні клієнти керуються чистою вигодою та бажанням отримати якомога більший прибуток за якомога менших витрат	Горизонтальна комунікація – зв'язки через сусідні бізнеси, через знайомих. Виставки нової техніки та технологій.	Швидкість. Функціональність. Простота.	Донести до потенційних клієнтів інформацію про існування продукту та про переваги способу обприскування за допомогою дронів перед класичними методами. Демонстрація можливостей системи	«Новий погляд на старі речі»

Висновки до розділу 5

В цьому розділі проведено розробку стартап-проєкту, яка включає такі етапи як: генерування ідеї проєкту її технологічний аудит, аналіз можливостей виходу цієї ідеї на ринок, розробку ринкової та маркетингової стратегій проєкту.

Виходячи з результатів аналізу ринку, можна зробити висновок, що стартап-проєкт має усі шанси на успіх, враховуючи фактори малої конкуренції, попиту на

технології, які б забезпечили зменшення витрат та збільшення прибутків та короткі терміни окупності проєкту. Оскільки на ринку майже немає прямих конкурентів, впровадження може бути доволі простим. Також наявність відносно великої кількості аграрних підприємств в Україні забезпечує певний рівень конкуренції між ними, що стимулює підприємців шукати способи підвищення прибутку шляхом залучення нових технологій.

Також у цьому розділі було визначено, що випуск сервісу з обмеженим функціоналом, з подальшим його вдосконаленням, є найбільш прийнятною альтернативою впровадження.

Подальший же розвиток проєкту є не тільки доцільним, а й обов’язковим, оскільки тільки таким чином вдасться завоювати свій сегмент ринку, постійно доповнюючи сервіс новим функціоналом та вносити зміни в уже існуючий, дослухаючись до думки користувачів.

ВИСНОВКИ

У цій роботі було розроблено інформаційну систему з підтримки обприскування сільськогосподарських полів за допомогою БПЛА.

Спочатку було проаналізовано сучасні методи обприскування, визначено їхні переваги та недоліки. В тому ж розділі було розглянуто метод обприскування за допомогою БПЛА, а також його порівняльний аналіз із традиційними методами обприскування. Під кінець першого розділу було визначено проблематику та сформульовано задачу цієї роботи – розроблення інформаційної системи, яка б дала змогу налаштування обприскування полів за допомогою різних дронів.

Наступним кроком у цій роботі було розглянуто різні існуючі на ринку рішення, проаналізовано їхні переваги та недоліки. На основі цього аналізу було сформовано вимоги до системи, яка була розроблена у цій роботі.

Після визначення вимог до системи відбулося її проєктування. На цьому етапі було розглянуто різні підходи до формування архітектури системи, та обрано найкращий з них. Після цього було сформовано опис функціональної складової системи, в якому зазначено основний функціонал, з яким взаємодіятимуть користувачі. Також в цьому розділі було визначено послідовність взаємодії користувачів із системою. Спираючись на усі ці пункти, було обрано засоби розробки системи, базу даних та розглянуто методи, які могли б стати у нагоді при реалізації алгоритму побудови оптимальної траєкторії обходу покриття.

На основі визначених вимог, архітектури, функціональності, порядку взаємодії, засобів розробки системи, інформації про різні види баз даних, та розглянутих методів побудови траєкторії було реалізовано інформаційну систему, яка відповідає головним завданням, які перед нею ставилися: можливість побудови оптимальної траєкторії обходу покриття, досягнення високої швидкодії та реалізація можливості автономного керування дронами. У четвертому розділі описується власне розроблена система: її загальна структура та структура її

компонентів й алгоритмів (у тому числі алгоритму побудови оптимальної траєкторії), а також її інтерфейси користувача.

Останнім розділом є розроблення стартап-проєкту, в якому було проаналізовано можливості виводу розробленої інформаційної системи на ринок. В рамках цієї задачі було описано ідею проєкту, проведено технологічний аудит цієї ідеї, проаналізовано ринкові можливості, та на кінець розроблено ринкову та маркетингову стратегії

Розроблена в цій роботі інформаційна система може якісно змінити сільськогосподарський сектор України в кращу сторону, оскільки обприскування за допомогою БПЛА є менш витратним і водночас більш ефективним. Окрім цього система може надати аграрним підприємствам можливість легко долучитися до використання БПЛА у своїх господарствах, без необхідності довго розбиратися в темі та займатися налаштуванням дронів самотужки.

Новизною ж розробленої системи є використання оптимального алгоритму пошуку траєкторії обходу покриття, що поєднує обхід робочої області за сіткою квадратів та вздовж її периметра для досягнення якнайбільшої площі покриття. Алгоритм використовує Чебишевську, мангетенську, вертикальну та горизонтальну евристичні функції, а також алгоритм знаходження найближчої непройденої комірки A* для знаходження оптимальної траєкторії по сітці.

Ще однією перевагою розробленої системи над існуючими аналогами є використання NoSQL бази даних, а саме DynamoDB, яка заточена на досягнення високої швидкодії під час роботи з величезними масивами даних, якими, до прикладу, є геопросторові координати. Підлаштована під запити DynamoDB значно покращує швидкодію системи, та в рази поліпшує користувацький досвід.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Difference between UAV drone spraying and traditional spraying [Електронний ресурс] – Режим доступу до ресурсу : https://www.dronefromchina.com/new/new-65-813.html (дата звернення : 08.12.2023).

2. Unmanned Aerial Vehicles (UAV), Unmanned Aerial Systems (UAS), AND Autonomous Drones : What's The Difference? [Електронний ресурс] – Режим доступу : https://www.missiongo.io/unmanned-aerial-vehicles-uav-unmanned-aerial-systems-uas-and-autonomous-drones-whats-the-difference/ (дата звернення : 10.12.2023).

3. AG-272 Product Page [Електронний ресурс] – Режим доступу до ресурсу : https://www.hyl.io/ag-272-product-page (дата звернення : 09.12.2023).

4. Hylio Drone Configurator [Електронний ресурс] – Режим доступу до ресурсу : https://configurator.hyl.io/#/configurator (дата звернення : 09.12.2023).

5. Причіпні обприскувачі купити в Україні - "Астарта Т" [Електронний ресурс] – Режим доступу до ресурсу : https://astartat.com.ua/ua/tehnika/Vnesenie-udobreniy/Opryiskivateli/pricepnye#/sort=p.price/order=DESC/limit=12 (дата звернення : 09.12.2023).

6. The Iron Triangle : Quality, Speed, Price - Pick Two [Електронний ресурс] – Режим доступу до ресурсу : https://www.unnatbak.com/blog/the-iron-triangle (дата звернення : 09.12.2023).

7. Morales-Rodríguez P. A. A comparison between conventional sprayers and new UAV sprayers : A study case of vineyards and olives in Extremadura / P. A. Morales-Rodríguez, E. Cano Cano, J. Villena, J. A. López-Perales // Agronomy. – 2022. – Т. 12, № 6 – С. 1307.

8. Panjaitan S. D. A Drone Technology Implementation Approach to Conventional Paddy Fields Application / S. D. Panjaitan, Y. S. K. Dewi, M. I. Hendri, R. A. Wicaksono, H. Priyatman // IEEE Access. – 2022. – Т. 10 – С. 120650-120658.

9. Rosedi, F. A. Comparative efficacy of drone application in chemical spraying at Paddy Field mada kedah. / F. A. Rosedi, S. M. Shamsi // IOP Conference Series: Earth and Environmental Science. – 2022 – Т. 1059, № 1 – С. 012002.

10. History of Drones in Agriculture. 2023. [Електронний ресурс] – Режим доступу : https://semantictech.in/blogs/history-of-drones-in-agriculture/ (дата звернення : 10.12.2023).

11. Agriculture Drone Crop Spraying UK. [Електронний ресурс] – Режим доступу : https://dronesurveyservices.com/agriculture-drone-crop-spraying/ (дата звернення : 10.12.2023).

12. The Advantages of Microservices vs Monolithic Architectures. [Електронний ресурс] – Режим доступу : https://levelup.gitconnected.com/the-advantages-of-microservices-vs-monolithic-architectures-94ce25ae3fd (дата звернення : 11.12.2023).

13. Microservices Architecture for Enterprise Large-Scaled Application. [Електронний ресурс] – Режим доступу : https://medium.com/design-microservices-architecture-with-patterns/microservices-architecture-for-enterprise-large-scaled-application-825436c9a78a (дата звернення : 11.12.2023).

14. Difference between Client /Server and Distributed DBMS. [Електронний ресурс] – Режим доступу : https://www.geeksforgeeks.org/difference-between-client-server-and-distributed-dbms/ (дата звернення : 11.12.2023).

15. What is Event-driven Architecture? [Електронний ресурс] – Режим доступу : https://www.tibco.com/reference-center/what-is-event-driven-architecture (дата звернення : 12.12.2023).

16. Посвістак В. С. Клієнт-серверна архітектура та її використання при розробці програмного забезпечення / В. С. Посвістак, Т. І. Демківська // Інформаційні технології в науці, виробництві та підприємництві : збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп’ютерних наук та технологій / за заг. наук. ред. В. Ю. Щербаня. – Київ : Освіта України ; ФОП Маслаков, 2020. – С. 78-81.

17. Overview of ASP.NET Core. [Електронний ресурс] – Режим доступу : https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0 (дата звернення : 13.12.2023).

18. ASP.NET Core 8 Pros and Cons. [Електронний ресурс] – Режим доступу : https://ukad-group.com/blog/aspnet-core-8-pros-and-cons/ (дата звернення : 13.12.2023).

19. MVC Framework - Introduction. [Електронний ресурс] – Режим доступу : https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (дата звернення : 13.12.2023).

20. OpenLayers - Welcome. [Електронний ресурс] – Режим доступу : https://openlayers.org/ (дата звернення : 14.12.2023).

21. Introduction to Node.js. [Електронний ресурс] – Режим доступу : https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення : 14.12.2023).

22. Understanding What is Webpack. [Електронний ресурс] – Режим доступу : http://parseobjects.com/understanding-what-is-webpack/ (дата звернення : 14.12.2023).

23. Khan, W. SQL and NoSQL database software architecture performance analysis and assessments – A systematic literature review. / W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, B. Luo // Big Data and Cognitive Computing. – 2023 – Т. 7, № 2 – С. 97.

24. Sahatqija K. Comparison between relational and NOSQL databases. / K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi, F. Ismaili // 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – 2018 – С. 216-221.

25. Nayak A. Type of NOSQL Databases and its Comparison with Relational Databases. / A. Nayak, A. Poriya, D. Poojary // International Journal of Applied Information Systems. – 2013 – Т. 5, № 4 – С. 16-19.

26. Tudorica B. G. Type of NOSQL A comparison between several NoSQL databases with comments and notes. / B. G. Tudorica, C. Bucur // RoEduNet

International Conference 10th Edition : Networking in Education and Research. – 2011 – C. 1-5.

27. Tang E. Performance Comparison between Five NoSQL Databases. / E. Tang, Y. Fan // 7th International Conference on Cloud Computing and Big Data (CCBD). – 2016 – C. 105-109.

28. Kalid S. Big-data NoSQL databases : A comparison and analysis of “Big-Table”, “DynamoDB”, and “Cassandra”. / S. Kalid, A. Syed, A. Mohammad, M. N. Halgamuge // IEEE 2nd International Conference on Big Data Analysis (ICBDA). – 2017 – C. 89-93.

29. Fevgas, G. Type of NOSQL Coverage path planning methods focusing on energy efficient and cooperative strategies for Unmanned Aerial Vehicles. / G. Fevgas, T. Lagkas, V. Argyriou, P. Sarigiannidis // Sensors. – 2022 – Т. 22, № 3 – С. 1235.

ДОДАТОК К

Посилання на GitHub з кодом системи

https://github.com/Nik-O-Lyandia/Field_spraying_ASP.NET_MVC