НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_комп'ютерних наук_____
(назва кафедри)

_____/Голуб Б.Л. /
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна управляюча система обліку таксації лісових угідь»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

_____д.е.н., проф._____ _____ _Руденський Р.А._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____к.т.н., с.н.с._____ _____ Боярінова Ю.Є._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ ___Пономаренко Артем Сергійович___
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_
(назва кафедри)
_к.т.н., доцент_____ _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)

"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

_____Пономаренко Артем Сергійович_____
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи Інформаційна управляюча система обліку таксації лісових угідь _____

затверджена наказом ректора НУБіП України від "22" квітня 2025р. №661 «С»

Термін подання завершеної роботи на кафедру__2025.05.23_____
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Отримання оцінки щодо коштовності лісу, оподаткування земельних ділянок____

Перелік питань, які потрібно розробити:

Вступ і постановка проблеми, мета та сфера застосування, огляд літератури, методологія, архітектура та дизайн системи, впровадження, оцінка та результати_____

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____Боярінова Ю.Є.__
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ ___Пономаренко А.С._____
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Ліси відіграють вирішальну роль у підтримці екологічної рівноваги, забезпечуючи середовища проживання різноманітних видів і пропонуючи різні екосистемні послуги, необхідні для добробуту людини. Таким чином, управління та збереження лісових ресурсів є найважливішими. Одним із важливих аспектів управління лісами є оподаткування, яке забезпечує стале використання лісових земель, водночас приносячи дохід державним органам, відповідальним за управління ними.

Традиційно оподаткування лісових земель було трудомістким і тривалим процесом, який часто покладався на ручну оцінку, яку проводили експерти лісового господарства. Однак прогрес у технології проклав шлях до більш ефективних і точних методів оцінки лісистих площ. У цьому контексті перспективним рішенням є розробка програмного забезпечення для автоматизованої системи таксації лісових земель у польових умовах[2].

Ця бакалаврська кваліфікаційна робота спрямована на дослідження концептуалізації, дизайну та впровадження такого програмного забезпечення, яке оптимізує процес оподаткування шляхом автоматизації та цифровізації. Завдяки інтеграції вдосконалених алгоритмів і геопросторових методів, це програмне забезпечення полегшує збір даних у режимі реального часу, аналіз і звітність безпосередньо з поля. Такі можливості не тільки підвищують ефективність і точність оцінки лісових земель, але й зменшують адміністративне навантаження на органи лісового господарства.

Метою цієї дипломної роботи є розробка програмного забезпечення для автоматизованої системи, спрямованої на спрощення та оптимізацію процесів таксації лісових земель, що проводяться в польових умовах. Використовуючи технологічні досягнення, програмне забезпечення прагне підвищити ефективність, точність і надійність оцінки лісистих територій для цілей оподаткування, зрештою сприяючи ефективному управлінню лісами та отриманню прибутку.

Завдання дослідження:

- провести комплексний огляд існуючих методологій і технологій, які використовуються в оподаткуванні лісових земель, висвітливши їх сильні сторони, обмеження та потенційні області для вдосконалення;
- проектування та розробка архітектури програмного забезпечення для автоматизованої системи, спеціально розробленої для вирішення проблем та вимог проведення оподаткування лісових земель у польових умовах;
- впровадження та ретельне тестування розробленого програмного забезпечення, забезпечуючи повну інтеграцію з існуючими системами геопросторових даних і системами оподаткування;
- оцінка продуктивності, ефективності і точності автоматизованої системи за допомогою порівняльного аналізу, експериментів і перевірки з традиційними методами оподаткування.

Об’єкт цього дослідження охоплює процеси та проблеми, пов’язані з оцінкою та оподаткуванням лісових земель у польових умовах. Зокрема, основна увага приділяється розробці та використанню програмного забезпечення в рамках автоматизованої системи для оптимізації та спрощення цих процесів. Вивчаючи застосування програмного забезпечення в оподаткуванні лісових земель, це дослідження спрямоване на підвищення точності оцінок, мінімізацію ручних зусиль і підвищення загальної ефективності управління лісовими ресурсами.

1 СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Розробка програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах стикається з різними викликами та можливостями в галузі управління та оподаткування лісового господарства.

Традиційні методи оподаткування в лісових районах є трудомісткими та схильними до невідповідностей, особливо коли вони проводяться в польових умовах. Ці умови, які характеризуються пересіченою місцевістю та несприятливими погодними умовами, ще більше ускладнюють точність і своєчасність нарахування податків[1].

Точна оцінка лісистих площ потребує вичерпних даних про такі фактори, як землекористування, породний склад дерев та екологічне значення. Збір і аналіз даних вручну забирає багато часу та може призвести до помилок, що підкреслює потенційні переваги автоматизованих систем.

Геопросторові технології, включаючи ГІС та дистанційне зондування, пропонують цінні інструменти для збору та обробки просторових даних, що стосуються оподаткування лісових земель. Інтеграція цих технологій в автоматизовану систему може забезпечити детальне уявлення про характеристики лісу, підвищуючи точність податкових оцінок.

Моніторинг лісових масивів у режимі реального часу має важливе значення для ефективного управління, що дозволяє своєчасно реагувати на такі проблеми, як незаконні рубки та лісові пожежі. Автоматизована система, оснащена можливостями дистанційного зондування, сприяє безперервному моніторингу, сприяючи проактивним методам управління.

Забезпечення зручності використання та доступності програмного забезпечення має вирішальне значення для його впровадження та

ефективності в польових умовах. Зручні інтерфейси, адаптовані до потреб фахівців лісового господарства, а також програми навчання та підтримки можуть сприяти успішному впровадженню в різних регіонах і юрисдикціях.

Розробка програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах дає можливість подолати виклики, пов’язані з традиційними методами, і підвищити ефективність і сталість методів управління лісами. Використовуючи технологію, інтегруючи геопросторові інструменти та віддаючи пріоритет зручності використання, ці зусилля спрямовані на оптимізацію процесів оподаткування та сприяння ефективному управлінню лісовими ресурсами[2].

У дипломній роботі досліджується складна розробка програмного забезпечення, спеціально розробленого для автоматизованої системи, спрямованої на полегшення складного процесу оподаткування лісових земель, особливо в складних польових умовах. Оподаткування лісових земель є критично важливим компонентом отримання прибутку та сталого управління ресурсами, включаючи оцінку та стягнення податків на лісисті території. Робота в польових умовах створює унікальні виклики, включаючи пересічену місцевість і змінну погоду, що вимагає програмного забезпечення, здатного збирати та аналізувати дані в реальному часі.

Центральне місце в цих зусиллях займає автоматизація, яка оптимізує різні завдання, такі як обробка даних і звітність, тим самим зменшуючи кількість помилок і прискорюючи процеси прийняття рішень. Використовуючи геопросторову технологію, програмне забезпечення інтегрує та аналізує просторові дані для полегшення точного оподаткування. У дисертації значний акцент приділяється розробці програмного забезпечення, зосереджуючись на створенні зручних для користувача інтерфейсів, надійних алгоритмів і повної інтеграції з існуючими базами даних і геопросторовими інструментами.

Сучасним системам оподаткування лісових земель часто бракує повної інтеграції можливостей збору та аналізу даних у реальному часі. Хоча деякі

системи можуть автоматизувати певні аспекти процесу оподаткування, такі як введення даних або обчислення, вони все ще можуть покладатися на ручні або напівручні методи збору просторових даних і проведення оцінок на місці в польових умовах. Така залежність від традиційних методів може призвести до затримок, неточностей і неефективності процесу оподаткування[3].

Крім того, багато існуючих систем можуть не повністю використовувати потенціал передових технологій, таких як штучний інтелект (AI), машинне навчання (ML) і хмарні обчислення. Ці технології можуть уможливити більш складний аналіз даних, прогнозне моделювання та можливості підтримки прийняття рішень, що зрештою підвищить точність і ефективність оподаткування лісових земель.

Крім того, доступність і зручність використання сучасних систем можуть бути обмежені, особливо у віддалених районах або регіонах з обмеженими ресурсами, де може бути брак підключення до Інтернету та технічного досвіду. Удосконалення інтерфейсу користувача та забезпечення надійних механізмів навчання та підтримки можуть допомогти усунути це обмеження та забезпечити ширше впровадження автоматизованих систем оподаткування лісових земель.

Підсумовуючи, сучасним системам оподаткування лісових земель може бракувати повної інтеграції можливостей збору та аналізу даних у реальному часі, передових технологічних функцій, а також покращеної доступності та зручності використання. Усунення цих прогалин може призвести до більш ефективних, точних і доступних процесів оподаткування лісових земель, що зрештою сприятиме кращому управлінню та збереженню лісових ресурсів.

Підвищуючи ефективність і точність процесів оподаткування лісових земель, дослідження має на меті сприяти покращенню практики управління та сталого використання ресурсів. Кінцева мета полягає в тому, щоб надати лісовим органам повноважний набір інструментів, який оптимізує процедури оподаткування, забезпечуючи тим самим ефективне управління лісовими ресурсами в умовах мінливих екологічних та економічних викликів[4].

1.2 Аналіз вимог до програмної системи

Аналіз вимог до програмної системи для автоматизованої системи оподаткування лісових земель у польових умовах передбачає розуміння основних функціональних можливостей, необхідних для успішної розробки та впровадження.

Програмне забезпечення має сприяти збору даних у режимі реального часу з польових оцінок, включаючи лісовий покрив, види дерев та екологічні характеристики. Інтеграція з геопросторовими технологіями, такими як географічні інформаційні системи (ГІС) і дистанційне зондування, має вирішальне значення для збору, зберігання та аналізу просторових даних, що стосуються оподаткування лісових земель.

Ефективна автоматизація повторюваних завдань, таких як введення даних і розрахунок податкових нарахувань, життєво важлива для продуктивності та точності. Включення передових алгоритмів і аналітичних методів забезпечує точні оцінки податків на основі геопросторових даних і просторового аналізу[5].

Програмне забезпечення повинно ефективно працювати в складних польових умовах, підтримуючи збір даних і синхронізацію в автономному режимі. Зручний інтерфейс з інтуїтивно зрозумілою навігацією та настроюваними робочими процесами покращує зручність використання та сприяння серед професіоналів лісового господарства.

Заходи безпеки повинні захищати конфіденційні дані, зібрані під час оцінки лісових земель, забезпечуючи дотримання відповідних правил захисту даних. Масштабованість і гнучкість дозволяють адаптуватися до мінливих вимог і технологічного прогресу з часом.

Функціональні вимоги:

1. збір даних у режимі реального часу: програмне забезпечення має забезпечувати збір даних у режимі реального часу під час польових оцінок, включаючи інформацію про лісовий покрив, види дерев, віковий розподіл та екологічні характеристики;

2. геопросторова інтеграція: інтеграція з геоінформаційними системами (ГІС) і технологіями дистанційного зондування необхідна для збору, зберігання та аналізу просторових даних, що стосуються оподаткування лісових земель;

3. автоматизоване нарахування податків: програмне забезпечення має автоматизувати розрахунок податкових нарахувань на основі зібраних даних і попередньо визначених алгоритмів, забезпечуючи точність і узгодженість процесів оподаткування;

4. функціональність в автономному режимі: враховуючи часто віддалені та складні польові умови, програмне забезпечення має підтримувати можливості збору даних в автономному режимі, дозволяючи користувачам проводити оцінювання без підключення до Інтернету та синхронізувати дані пізніше;

5. інтерфейс користувача: зручний інтерфейс із інтуїтивно зрозумілою навігацією, чіткою візуалізацією даних і настроюваними робочими процесами є важливим для полегшення використання та впровадження серед професіоналів лісового господарства.

Нефункціональні вимоги:

1. продуктивність: програмне забезпечення має бути здатним ефективно обробляти великі обсяги даних, своєчасно обробляти обчислення та аналізи, щоб відповідати вимогам польових оцінок;

2. надійність: система має бути надійною та стійкою, здатною стабільно працювати в різноманітних польових умовах і протистояти потенційним збоям, таким як перебої в електропостачанні або збої в мережі[5];

3. безпека: необхідно вжити заходів безпеки, щоб захистити конфіденційні дані, зібрані під час оцінювання, включаючи протоколи

шифрування, механізми автентифікації користувачів і засоби контролю доступу;

4. масштабованість: програмне забезпечення має бути масштабованим, щоб відповідати зростаючим обсягам даних і вимогам користувачів з часом без шкоди для продуктивності чи зручності використання;

5. сумісність: сумісність із низкою пристроїв і операційних систем є важливою для забезпечення доступності та зручності використання на різних платформах і середовищах;

6. відповідність нормативним вимогам: програмне забезпечення має відповідати відповідним нормам захисту даних і галузевим стандартам, забезпечуючи етичне поводження з конфіденційною інформацією, зібраною під час оцінки лісових земель.

Виконуючи ці вимоги до системи програмного забезпечення, розробники можуть створити надійну автоматизовану систему, адаптовану до конкретних потреб оподаткування лісових земель у польових умовах, ефективно підтримуючи практики сталого управління лісами.

1.3 Моделювання предметної області

UML, або Уніфікована мова моделювання, служить стандартизованою мовою візуального моделювання, яка використовується в розробці програмного забезпечення та проектуванні систем. Він пропонує набір графічних позначень і символів, що дозволяє розробникам, дизайнерам і зацікавленим сторонам створювати візуальні представлення різних аспектів системи. Ці візуалізації допомагають зрозуміти, проаналізувати та задокументувати архітектуру системи, компоненти, зв’язки та процеси[6].

UML знаходить застосування в моделюванні різних типів систем, включаючи програмні додатки, бази даних, бізнес-процеси та організаційні структури. Він забезпечує стандартизований підхід для створення діаграм і

моделей, які сприяють ефективній комунікації та співпраці між членами команди, які беруть участь у розробці та обслуговуванні системи.

Діаграми UML охоплюють кілька типів, включаючи діаграми класів, що представляють статичну структуру системи, діаграми варіантів використання, що описують функціональні вимоги та взаємодію між акторами (користувачами) і системою, діаграми послідовності, що ілюструють послідовність взаємодій і повідомлень, якими обмінюються об’єкти або компоненти протягом певного часу. , діаграми діяльності, що ілюструють робочий процес і керування потоком дій або процесів у системі, діаграми кінцевих автоматів, що моделюють стани, події та переходи системи чи об’єкта, діаграми компонентів, що зображують фізичні або логічні компоненти системи та їхні зв’язки, і діаграми розгортання, що описують фізичне розгортання програмних компонентів на апаратних вузлах або пристроях.

Ці діаграми UML служать візуальними представленнями архітектури та поведінки системи, допомагаючи зацікавленим сторонам краще розуміти, аналізувати та повідомляти про дизайн і функціональність системи. UML надає стандартизовану мову, яка сприяє чіткому та короткому спілкуванню між різними зацікавленими сторонами, залученими до проектів розробки програмного забезпечення[7].

При побудові моделей складних систем в UML, принципи розділення підсистеми можуть керувати процесом. Розробка системи з підсистемами допомагає ефективно керувати складністю. Підсистеми можуть являти собою функціональні блоки або логічні групи компонентів. В якості альтернативи можна використовувати рівні абстракції. Різні рівні абстракції дозволяють представити систему на різних рівнях деталізації. Наприклад, використання загальних класів на вищому рівні, а потім деталізація їх у відповідні підкласи на нижчих рівнях.

Інший принцип передбачає використання зв’язків агрегації та композиції. Використання зв’язків агрегації та композиції додатково описує

зв’язки між компонентами системи. Агрегація передбачає, що один компонент містить або складається з інших компонентів, тоді як композиція вказує на міцні відносини власності між компонентами.

Ці принципи допомагають у побудові зрозумілих, гнучких і масштабованих моделей складних систем за допомогою UML. Вони роблять внесок у якісний аналіз і проектування системи, сприяючи спілкуванню між розробниками, дизайнерами та зацікавленими сторонами проекту.

В UML діаграма служить як графічне представлення, що зображує структурні, поведінкові або взаємодіючі аспекти системи або програмного додатку. UML пропонує різні типи діаграм, кожна зі своїм призначенням і нотацією, для візуалізації та передачі різних аспектів системи.

Діаграми UML служать засобом збору та передачі інформації про архітектуру, компоненти, зв’язки, поведінку та взаємодію елементів у системі. Вони використовуються для покращення розуміння, аналізу, проектування та документування програмних систем.

1.3.1 **Діаграма прецедентів**

Діаграма прецедентів – діаграма, що описує, який функціонал програмної системи, що розробляється, доступний кожній групі користувачів [10].

На діаграмі основними елементами є актори та варіанти використання.

Актор представляє логічну роль, яка бере участь у взаємодії з варіантами використання або сутностями, такими як системи, підсистеми або класи. Акторами можуть бути люди або інші системи, підсистеми або класи, які існують поза сутністю. Графічно актор зображений у вигляді фігурки.

Варіант використання описує послідовність подій, включаючи варіанти, які виконує система, що призводить до спостережуваного результату для актора. Він представляє поведінку суб'єкта, описуючи взаємодію між

акторами та системою. Варіант використання показує не те, як досягається конкретний результат, а радше те, що виконується. Варіанти використання позначаються просто еліпсами, а всередині їхня назва.

Розроблена діаграма варіантів використання зображена на малюнку 1.

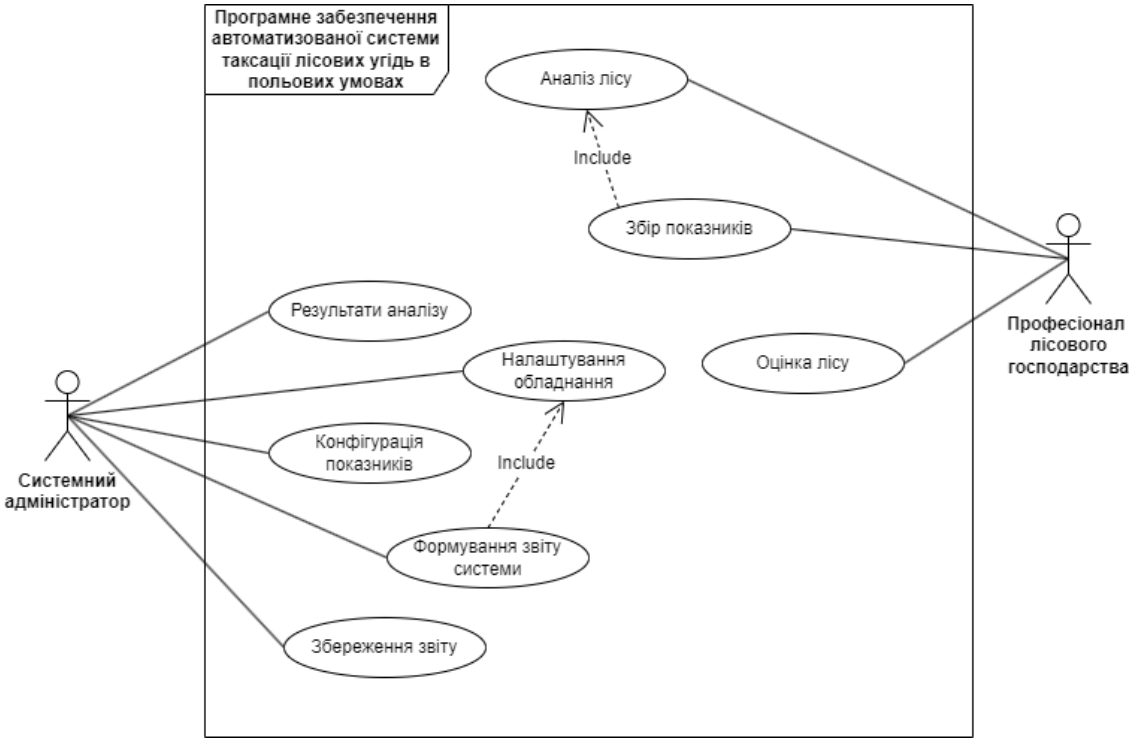

Рис. 1 Діаграма прецедентів

Створена діаграма прецедентів містить 2 актора:

- “Системний адміністратор”;
- “Професіонал лісового господарства”.

Актор «Системний адміністратор» включає такі прецеденти:

- конфігурація показників;
- налаштування обладнання;
- результати аналізу;
- формування звіту системи;
- збереження звіту.

Актор «Професіонал лісового господарства» включає такі прецеденти:

- аналіз лісу;
- збір показників;
- оцінка лісу.

Прецеденти певним чином залежать одне від одного.

Прецедент “Формування звіту системи” доповнюється іншим таким прецедентом як “Налаштування обладнання”. Також прецедент “Збір показників” доповнюється прецедентом “Аналіз лісу”.

Розглянемо детальніше вищеописані прецеденти.

Назва прецеденту використання: Аналіз лісу

Актор: Професіонал лісового господарства

Опис: цей сценарій використання передбачає проведення детального аналізу лісових масивів за допомогою аналітичних інструментів і алгоритмів програмного забезпечення, щоб отримати уявлення про різні аспекти управління лісами та оподаткування.

Передумова: Професіонал лісового господарства увійшов до системи програмного забезпечення та отримав доступ до модуля аналізу лісу.

Основний потік:

1. професіонал лісового господарства вибирає для аналізу конкретну лісову ділянку або регіон із доступних опцій, представлених в інтерфейсі програмного забезпечення;
2. система отримує відповідні набори даних, включаючи супутникові зображення, топографічні карти, класифікацію ґрунтового покриву та історичні записи про управління лісами для вибраної території;
3. професіонал лісового господарства налаштовує параметри аналізу, вказуючи такі критерії, як тип лісу, віковий розподіл, індекси біорізноманіття та екологічні показники, які слід враховувати під час аналізу;
4. програмне забезпечення використовує розширені аналітичні алгоритми для обробки даних і створення вичерпних звітів і візуалізацій, що підсумовують характеристики лісу, стан здоров’я та екологічну цінність;

5. професіонал лісового господарства переглядає результати аналізу, інтерпретуючи висновки, щоб визначити закономірності, тенденції та проблемні області в межах лісистої території;

6. на основі результатів аналізу лісовий аналітик формулює рекомендації та стратегії для оптимізації методів управління лісами, включаючи заходи щодо збереження, збільшення біорізноманіття та сталого використання ресурсів;

7. професіонал лісового господарства експортує звіти та висновки аналізу в різні формати, такі як документи PDF або файли, сумісні з ГІС, для подальшого перегляду, презентації або інтеграції в плани управління лісами.

Підсумок: процес аналізу лісу завершено, і професіонал лісового господарства отримав цінні відомості та рекомендації для інформування щодо прийняття рішень і планування, пов’язаних з оподаткуванням та управлінням лісовими землями.

Альтернативний сценарій:

- якщо вибрана лісова ділянка не має достатнього охоплення даних або якості для аналізу, система запропонує аналітику лісового господарства вибрати альтернативну ділянку або уточнити параметри аналізу;
- у разі технічних проблем або помилок під час обробки даних система генерує повідомлення про помилки та реєструє інциденти для усунення несправностей системними адміністраторами.

Назва прецеденту використання: Оцінка лісу

Актор: Професіонал лісового господарства

Опис: цей варіант використання передбачає автоматизовану оцінку земельних ділянок, вкритих лісом, на основі різних факторів, таких як тип лісу, породний склад дерев, віковий розподіл, екологічне значення та показники ринкової вартості.

Передумова: Професіонал лісового господарства увійшов до системи програмного забезпечення та отримав доступ до модуля оцінки лісу.

Основний потік:

1. професіонал лісового господарства вибирає лісову земельну ділянку з наявного списку властивостей, щоб розпочати процес оцінки;

2. система отримує відповідні дані, пов’язані з вибраною земельною ділянкою, включаючи геопросторові дані, записи лісо утворення, екологічні оцінки та індекси ринкових цін, з інтегрованих баз даних і зовнішніх джерел;

3. професіонал лісового господарства переглядає зібрані дані та перевіряє їх точність і повноту, гарантуючи наявність усієї відповідної інформації, необхідної для оцінки;

4. програмне забезпечення використовує алгоритми та моделі оцінки, враховуючи такі фактори, як тип лісу, породний склад дерев, віковий розподіл, екологічну цінність, правила землекористування та ринкові тенденції, для розрахунку оціночної вартості земельної ділянки, вкритої лісом;

5. система генерує повний звіт про оцінку, у якому представлено розраховану вартість земельної ділянки разом із детальною інформацією про фактори, що впливають на процес оцінки;

6. професіонал лісового господарства переглядає звіт про оцінку, аналізуючи результати та враховуючи будь-які додаткові фактори чи коригування, які можуть знадобитися на основі місцевого законодавства чи конкретних характеристик власності;

7. на підставі результатів оцінки професіонал лісового господарства приймає обґрунтовані рішення щодо нарахування податку на нерухомість земельної ділянки, вкритої лісом, забезпечуючи дотримання податкового законодавства та нормативних актів, одночасно відображаючи справедливу ринкову вартість майна;

8. професіонал лісового господарства завершує звіт про оцінку та оновлює відповідні податкові записи та бази даних оціночною вартістю лісової земельної ділянки для цілей оподаткування.

Альтернативний сценарій:

- якщо в зібраній інформації є розбіжності або відсутні дані, система попереджає податкового оцінювача, щоб він перевірив і усунув проблеми, перш ніж продовжити процес оцінки;
- у разі технічних помилок або збоїв у роботі системи під час розрахунків оцінки, система генерує повідомлення про помилку та сповіщає системних адміністраторів для вирішення, щоб забезпечити точні та надійні результати оцінки.

1.3.2 Діаграма послідовності.

Дана діаграма графічно демонструє порядок взаємодії певних об'єктів програми у годині. Як правило, в цій діаграмі демонструється, як користувачі (актори з діаграми варіантів використання) взаємодіють з іншими компонентами програми під час реалізації тих чи інших варіантів використання програми, та як при цьому взаємодіють інші компоненти програмної системи. Зазвичай одна діаграма послідовності присвячена опису одного з варіантів використання, зазначеного в Use-case діаграмі [12].

Діаграми послідовності є одним із способів формалізації сценаріїв використання. Її перевага заклепується в тому, що на ранніх стадіях опису сценаріїв можна з'ясувати склад взаємодіючих компонентів та описати потоки повідомлень від одних компонентів до інших. Ці компоненти та потоки повідомлень надалі будуть трансформовані у конкретні класи (об'єкти), методи цих об'єктів (якщо говорити термінологією мови Java). Відповідно, відразу з'ясовується і модель системи подій (Actions), які дані класи (об'єкти) будуть підтримувати та обробляти.

Діаграма послідовностей, зображена на рис. 2, містить такі об’єкти:

- професіонал лісового господарства;
- аналіз лісу;
- конфігурація;
- оцінка вартості лісу.

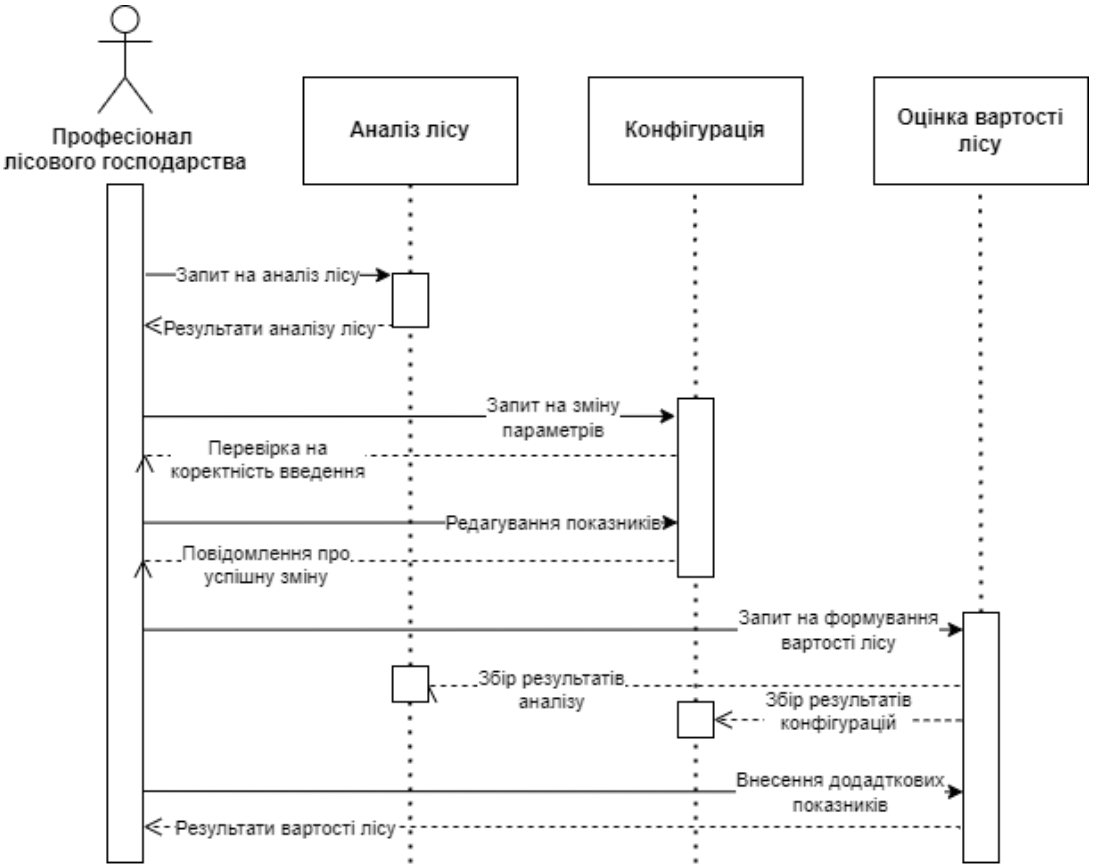

Рис. 2 Діаграма послідовності

Ця діаграма зображує послідовність взаємодії між професіоналом лісового господарства, автоматизованою системою та базою даних. Професіонал лісового господарства входить в систему, збирає польові дані та генерує податкові оцінки за допомогою системи, яка взаємодіє з базою даних для отримання просторових даних і правил оподаткування. Нарешті система обробляє дані та повертає податкову оцінку.

1.4 Огляд інформаційних джерел та існуючих рішень

Розглянемо існуючі рішення предметної області.

Програма інвентаризації та аналізу лісів (FIA), якою керує Лісова служба США (USFS), є важливою ініціативою, спрямованою на моніторинг та оцінку лісів країни. Це одна з наймасштабніших безперервних

інвентаризацій лісів у світі, яка надає безцінні дані для розуміння лісових екосистем, управління ресурсами та формування політики.

FIA збирає дані через широку мережу польових ділянок, розподілених по Сполучених Штатах, що охоплюють різні типи лісів, форми власності та умови. Висококваліфіковані польові бригади збирають стандартизовані дані про різні характеристики лісу, включаючи види дерев, розподіл розмірів, вікову структуру, показники здоров'я, біорізноманіття та зберігання вуглецю.

Основні елементи цієї програми охоплюють збір польових даних, ретельний аналіз, вичерпну звітність, постійний моніторинг і дослідницькі зусилля. Зібрані дані інформують про процеси прийняття рішень на різних рівнях управління, допомагаючи у формуванні політики, розподілі ресурсів і зусиллях щодо збереження[17].

Висококваліфіковані польові бригади збирають дані за допомогою стандартизованих протоколів відбору проб. Вони вимірюють дерева, оцінюють показники здоров'я лісу та записують різні атрибути лісової екосистеми.

Зібрані дані аналізуються для створення комплексних звітів, баз даних і наборів просторових даних. Ці результати дають цінну інформацію про стан і тенденції лісових ресурсів на місцевому, регіональному та національному рівнях.

Дані FIA підтримують постійні зусилля з моніторингу для відстеження змін у лісових умовах з часом. Він також служить основою для наукових досліджень на такі теми, як динаміка лісів, екосистемні послуги та вплив зміни клімату.

Дані FIA інформують про рішення щодо управління лісами, розробки політики та розподілу ресурсів на всіх рівнях управління. Це допомагає зацікавленим сторонам зрозуміти економічну, екологічну та соціальну

цінність лісів і визначити пріоритетність заходів щодо їх збереження та управління[17].

Забезпечуючи стандартизований і комплексний підхід до інвентаризації та аналізу лісів, FIA робить значний внесок у просування сталого управління лісами, збереження біорізноманіття та стійкості екосистем. Його надійні дані та наукові висновки служать основою для прийняття обґрунтованих рішень і розробки політики як у Сполучених Штатах, так і як модель для міжнародних ініціатив з управління лісами.

Інтерфейс додатку «FIA» зображено на рис.3.

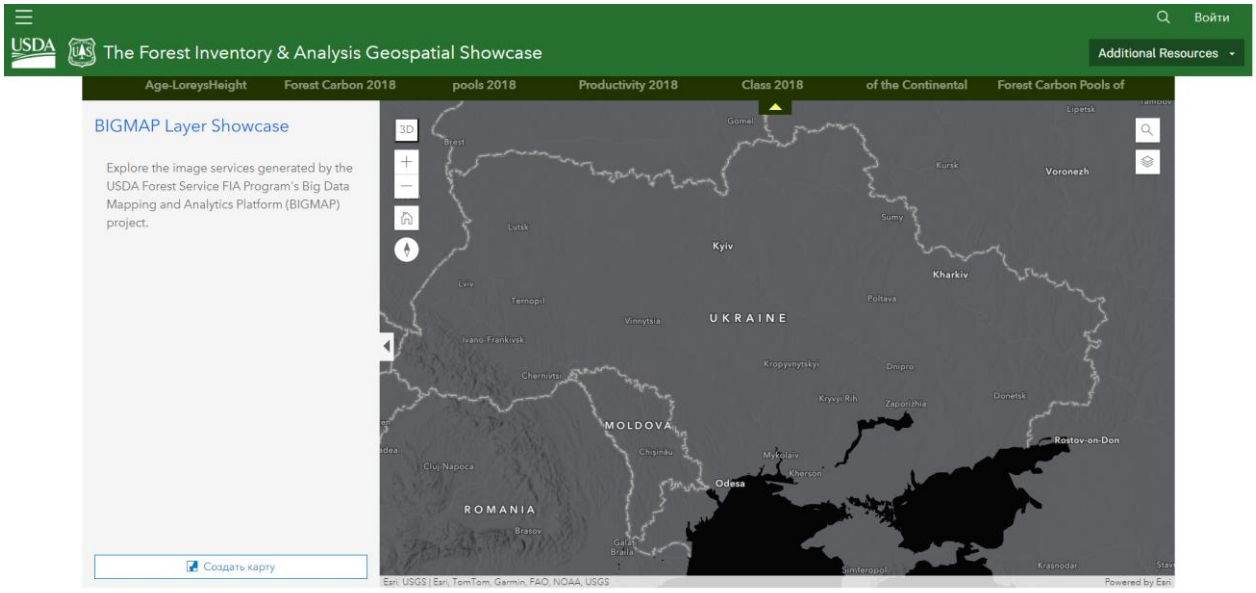

Рис. 3 Програмний додаток «FIA»

SilvAssist — це комплексне програмне рішення, призначене для оптимізації та вдосконалення процесів управління лісами та інвентаризації деревини. Розроблений для задоволення потреб лісників, землевпорядників і лісопромислових компаній, SilvAssist об’єднує передові технології для оптимізації лісогосподарських операцій і максимального підвищення ефективності.

SilvAssist полегшує збір польових даних за допомогою мобільних пристроїв, дозволяючи лісівникам ефективно збирати інформацію про види дерев, розмір, здоров’я та характеристики насаджень. Потім програмне

забезпечення аналізує ці дані, щоб надати цінну інформацію про стан лісів і ресурси деревини[18].

За своєю суттю SilvAssist забезпечує безперебійний збір і аналіз даних, дозволяючи лісівникам ефективно збирати інформацію про види дерев, розмір, стан здоров'я та характеристики насаджень за допомогою мобільних пристроїв. Потім програмне забезпечення використовує ці дані, щоб надати цінну інформацію про стан лісів і ресурси деревини.

Однією з видатних особливостей SilvAssist є його надійні можливості керування запасами, що дозволяє користувачам відстежувати обсяги деревини, темпи росту та графіки заготівлі. Ведучи точні інвентаризаційні записи, лісівники можуть приймати обґрунтовані рішення щодо заготівлі деревини, відновлення та довгострокового лісового планування.

Для планування та оптимізації лісозаготівлі SilvAssist надає модулі для розробки рецептів лісозаготівлі, планування лісозаготівельних операцій та оптимізації маршрутів видобутку деревини. Враховуючи такі фактори, як рельєф місцевості, об'єм деревини та екологічні обмеження, SilvAssist максимізує ефективність та сталість лісозаготівельних робіт.

Програмне забезпечення генерує настроювані звіти та документацію щодо відповідності, полегшуючи відповідність нормативним вимогам і вимогам до звітності зацікавлених сторін. Незалежно від того, документуєте продажі деревини, контролюєте стан лісу чи звітуєте про заходи по збереженню, SilvAssist спрощує процес звітування та забезпечує точність і підзвітність.

SilvAssist пропонує надійні можливості керування запасами, дозволяючи користувачам відстежувати обсяги деревини, темпи зростання та графіки заготівлі. Ведучи точні інвентаризаційні записи, лісівники можуть приймати обґрунтовані рішення щодо заготівлі деревини, відновлення та довгострокового лісового планування.

SilvAssist містить розширені інструменти картографування та візуалізації, які дозволяють користувачам створювати детальні карти лісових

масивів, включаючи межі насаджень, дорожню мережу та топографічні об’єкти. Ці карти допомагають візуалізувати лісові ресурси, визначити пріоритети управління та передавати плани зацікавленим сторонам.

SilvAssist містить модулі для планування та оптимізації лісозаготівлі, що дозволяє користувачам розробляти рецепти лісозаготівлі, планувати операції лісозаготівлі та оптимізувати маршрути видобутку деревини. Враховуючи такі фактори, як рельєф місцевості, об’єм деревини та екологічні обмеження, SilvAssist допомагає максимізувати ефективність та сталість лісозаготівельних робіт.

SilvAssist створює настроювані звіти та документацію щодо відповідності, сприяючи дотриманню нормативних вимог і вимогам щодо звітності зацікавлених сторін. Незалежно від того, документуєте продажі деревини, контролюєте стан лісу чи звітуєте про заходи по збереженню, SilvAssist спрощує процес звітування та забезпечує точність і підзвітність.

SilvAssist бездоганно інтегрується з іншими програмними системами лісового господарства та ГІС-платформами, забезпечуючи обмін даними та співпрацю між зацікавленими сторонами. Незалежно від того, чи працюєте ви з планами управління лісами, даними про рейси з деревиною чи наборами просторових даних, SilvAssist полегшує взаємодію та підвищує ефективність робочого процесу.

Загалом, SilvAssist є потужним інструментом для оптимізації методів управління лісами, удосконалення процесів прийняття рішень і сприяння сталим методам лісового господарства. Використовуючи інноваційні технології та надійні функції, SilvAssist дає змогу професіоналам лісового господарства ефективно управляти лісовими ресурсами та зберігати їх для майбутніх поколінь.

Інтерфейс додатку «SilvAssist» представлено на рис.4.

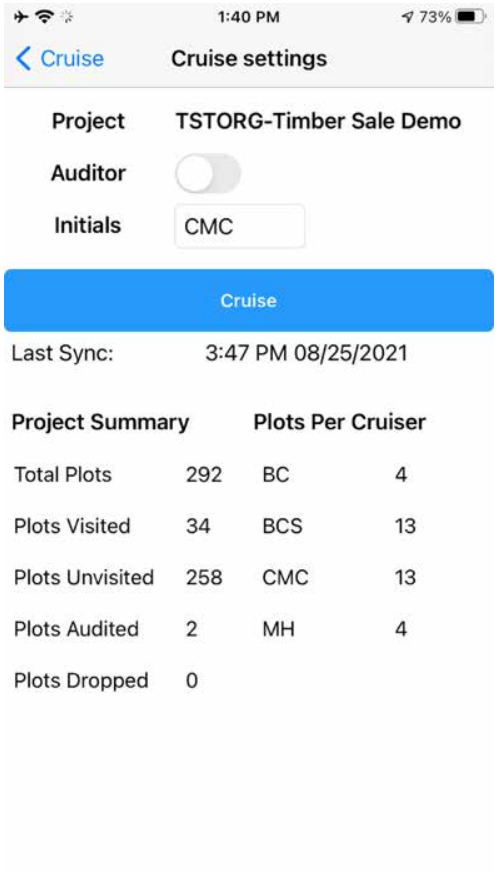

Рис. 4 Мобільний додаток “SilvAssist”

1.5 Постановка завдання

У дипломній роботі постановка проблеми передбачає виявлення та усунення проблем і недоліків сучасних методів таксації лісових земель у польових умовах. Це включає в себе кілька ключових аспектів:

Традиційні методи оподаткування лісових земель базуються на методах ручного збору даних та оцінки, які вимагають багато часу, праці та схильні до помилок. Автоматизація цих процесів може підвищити ефективність і точність.

Це охоплює кілька ключових аспектів:

- ручні процеси;
- точність і узгодженість даних;
- інтеграція технологій;
- відповідність нормативним вимогам;

- інтерфейс користувача та доступність;
- масштабованість і адаптивність.

Системи оподаткування лісових земель повинні відповідати законодавчим і нормативним вимогам, що регулюють використання землі, оподаткування та охорону навколишнього середовища. Важливо розробити автоматизовану систему, яка відповідає цим стандартам, забезпечуючи прозорість і підзвітність.

Зручність використання та доступність автоматизованої системи є критичними факторами для її успішного впровадження та впровадження. Розробка інтуїтивно зрозумілого інтерфейсу користувача, навчання та підтримка користувачів системи є важливими для досягнення максимальної ефективності[26].

Автоматизована система має бути масштабованою та адаптованою для адаптації до змін у практиці управління лісами, податковій політиці та технологічному прогресі з часом. Гнучкість проектування та архітектури системи необхідна для задоволення мінливих потреб і вимог.

Загалом формулювання проблеми в дисертації передбачає виявлення обмежень поточної практики оподаткування лісових земель, пропонування рішень шляхом розробки автоматизованої системи та оцінку її ефективності щодо підвищення ефективності, точності та дотримання нормативних стандартів.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

ERwin, потужний інструмент моделювання даних, який широко використовується в корпоративних середовищах, полегшує проектування та керування базами даних. Його зручний графічний інтерфейс дозволяє користувачам створювати, візуалізувати та документувати моделі баз даних на основі підходу Entity-Relationship (ER).

Щоб розпочати роботу з ERwin, користувачам необхідно встановити та налаштувати програмне забезпечення на своєму комп'ютері. Після встановлення вони можуть створювати нову модель даних в інтерфейсі ERwin, визначаючи сутності та їхні атрибути, що представляють об'єкти реального світу та їхні характеристики.

Ключовою особливістю ERwin є його здатність встановлювати зв'язки між сутностями, представляючи зв'язки або залежності між різними об'єктами. Користувачі можуть вказати характер цих зв'язків, включаючи обмеження участі та кардинальність[16].

Працюючи над моделлю даних, користувачі можуть вдосконалювати її, додаючи додаткові деталі, такі як первинні ключі, зовнішні ключі, індекси та обмеження. ERwin надає інструменти перевірки та аналізу для забезпечення структурної цілісності та дотримання найкращих практик. Користувачі також можуть генерувати сценарії SQL або оператори мови визначення даних (DDL), щоб створити фізичну схему бази даних на основі логічної моделі.

Документація є важливою частиною процесу моделювання, і ERwin пропонує можливості для створення повної документації для вашої моделі даних. Це включає діаграми сутності та зв'язку, словники даних і різні звіти.

Належна документація допомагає зрозуміти та підтримувати базу даних з часом.

Функції співпраці та керування версіями в ERwin дозволяють кільком користувачам працювати над тією самою моделлю даних одночасно. Це дозволяє відстежувати зміни, внесені в модель, і сприяє ефективній співпраці між членами команди.

ERwin спрощує процес проектування та керування базами даних, надаючи візуальний та інтуїтивно зрозумілий інтерфейс. Його функції моделювання, встановлення зв'язків, вдосконалення моделі, створення документації та можливості співпраці роблять його цінним інструментом для професіоналів баз даних у різних галузях.

Для побудови логічної моделі інформаційної системи була використана модель Entity-Relationship Diagram (ERD). Ця модель представляє інформацію як сутності з відповідними атрибутами та встановлює зв'язки між цими сутностями.

Основна мета створеної моделі – адекватно відображати потреби та вимоги системи. Кожна сутність у моделі представляє певний аспект інформаційної системи, а атрибути відображають характеристики цих сутностей[16].

Крім того, модель пов'язує сутності разом за допомогою зв'язків, які зображують зв'язки та асоціації між різними частинами інформаційної системи.

При розробці моделі ми дотримуємося двох основних принципів. По-перше, модель має бути конкретною, точно відображати характеристики та потреби системи. По-друге, модель має бути простою та зручною для майбутнього використання, забезпечуючи ясність та легкість роботи.

Створена логічна модель інформаційної системи відповідатиме цим вимогам, забезпечуючи наочне та зручне представлення інформаційної системи.

Логічна модель системи представлена на рисунку 5.

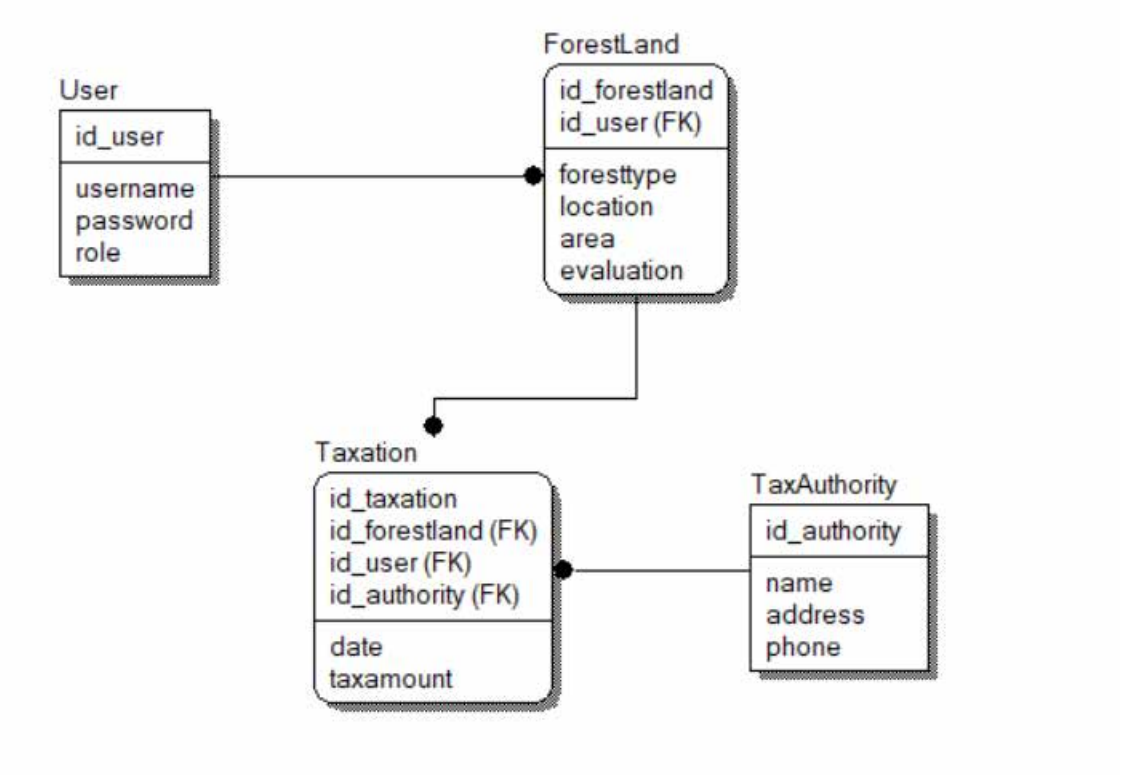

Рис. 5 ER-діаграма

Логічна модель складається з таких сутностей:

- “User” містить інформацію про користувача;;
- “Forestland” містить інформацію про лесову ділянку;
- “Taxation” містить інформацію про оцінку вартості лісу;
- “Taxauthority” містить інформацію про власника ділянки.

Сутність “User” має такі атрибути: id_user; username; password; role.

Сутність “Forestland” має такі атрибути: id_forestland; id_user; foresttype; location; area; evaluation.

Сутність “Taxation” має такі атрибути: id_taxation; id_forestland; id_user; id_authority; date; taxamount.

Сутність “Taxauthority” має такі атрибути: id_authority; name; address; phone.

Сутність «User» пов’язана з сутністю «ForestLand». Також сутність «ForestLand» зв’язана з сутністю «Taxation». І сутність “Taxation” зв’язана з сутностями “ForestLand” та “TaxAuthority”.

2.2 Діаграма класів та кооперацій

Діаграма класів визначає типи класів у системі та різноманітні статичні зв’язки між ними. Він описує атрибути, операції та обмеження, накладені на зв’язки між класами. Інтерпретація діаграми класів суттєво залежить від ракурсу: класи можуть представляти сутності домену (під час аналізу) або елементи програмної системи (під час проектування та впровадження).

Основні елементи включають класи та зв'язки між ними. Класи характеризуються атрибутами та операціями. Атрибути описують характеристики об'єктів класу. Більшість об’єктів у класі отримують свою індивідуальність через відмінності в атрибутах і зв’язках з іншими об’єктами. Однак можливі об’єкти з ідентичними значеннями атрибутів і зв’язками. Імена атрибутів мають бути унікальними в класі та можуть включати їх тип і значення за замовчуванням[18].

Операції представляють функції або перетворення. Вони можуть мати параметри та значення, що повертаються.

Типи зв’язків включають асоціацію, агрегацію та успадкування.

Спеціальна діаграма класів була створена з використанням об’єктно-орієнтованого підходу для цього веб-сайту (рис. 6).

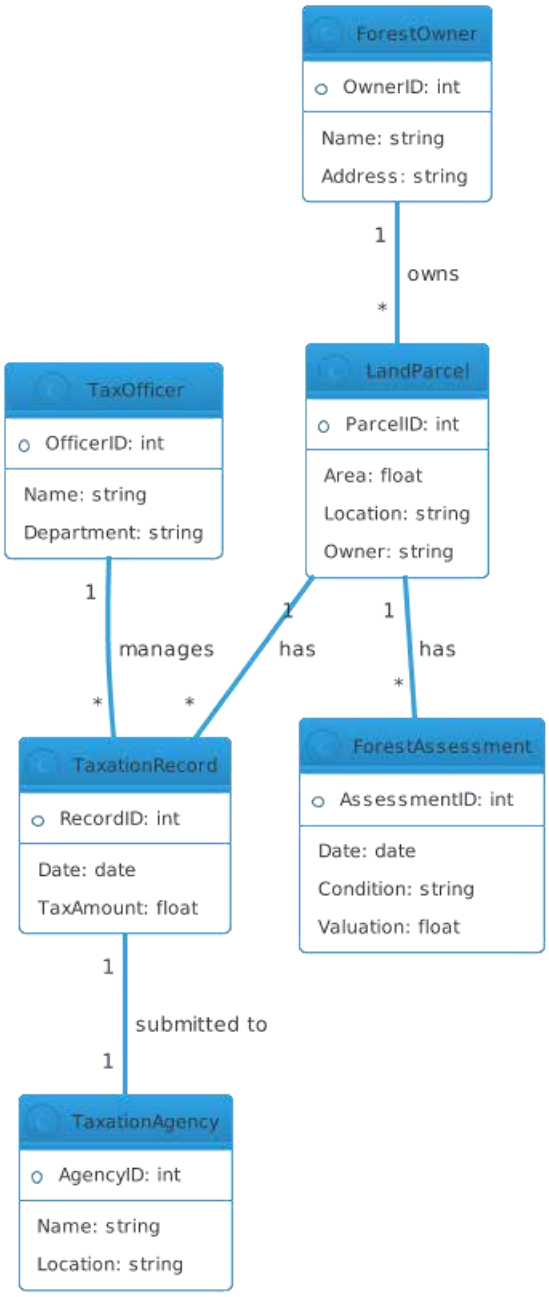

Рис. 6 Діаграма класів

ForestOwner – це земельна ділянка лісового фонду, що підлягає оподаткуванню. Він містить такі атрибути, як ParcelID, Area, Location і Owner. Кожна земельна ділянка може мати кілька таксаційних і таксаційних записів. Крім того, у власності одного лісовласника може бути декілька земельних ділянок.

Клас ForestAssessment представляє оцінку, проведену на земельній ділянці для оцінки її стану та вартості. Він містить такі атрибути, як AssessmentID, Date, Condition і Valuation. Кожна таксація лісу пов’язана з

окремою земельною ділянкою.

Клас TaxationRecord представляє запис про оподаткування конкретної земельної ділянки. Він містить такі атрибути, як RecordID, Date і TaxAmount. Кілька податкових записів можуть бути пов’язані з однією земельною ділянкою, і ними може керувати податковий офіцер. Крім того, кожен запис подається до єдиного податкового органу.

Клас TaxOfficer представляє податкового офіцера, відповідального за керування податковими записами. Він містить такі атрибути, як OfficerID, Name та Department. Декілька податкових записів може вести один податковий офіцер.

Клас ForestOwner представляє власника земельних ділянок лісового фонду. Він включає такі атрибути, як OwnerID, Name та Address. У власності одного лісовласника може перебувати декілька земельних ділянок.

Клас TaxationAgency представляє орган, відповідальний за оподаткування. Він містить такі атрибути, як AgencyID, Name та Location. Кожен податковий облік подається до єдиного податкового органу.

Діаграми кооперацій в UML пропонують графічне представлення того, як об’єкти взаємодіють для виконання системних функцій. Ці діаграми служать наочними посібниками для розуміння динамічної поведінки системи під час виконання. Об’єкти, зображені у вигляді прямокутників, співпрацюють за допомогою обміну повідомленнями, представленими посиланнями. Кожен об’єкт може брати на себе різні ролі та відповідальність у рамках співпраці, сприяючи загальній функціональності системи. Потік повідомлень між об’єктами проілюстровано для демонстрації послідовності взаємодій. Ця візуалізація дозволяє дизайнерам зрозуміти складні зв’язки та моделі зв’язку між об’єктами, тим самим полегшуючи проектування та аналіз складних систем.

Діаграми кооперацій особливо корисні на етапі проектування, допомагаючи у виявленні потенційних проблем і оптимізації продуктивності системи. Забезпечуючи чітке та стисле представлення об’єктної співпраці, ці

діаграми покращують спілкування між зацікавленими сторонами та спрощують процес розробки.

Для конкретного веб-сайту створено індивідуальні діаграми кооперацій (рис. 7-8).

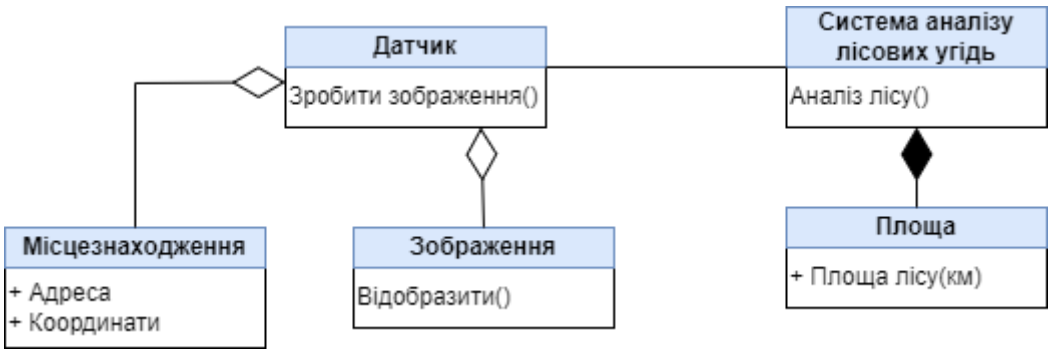

Рис. 7 Діаграма кооперацій

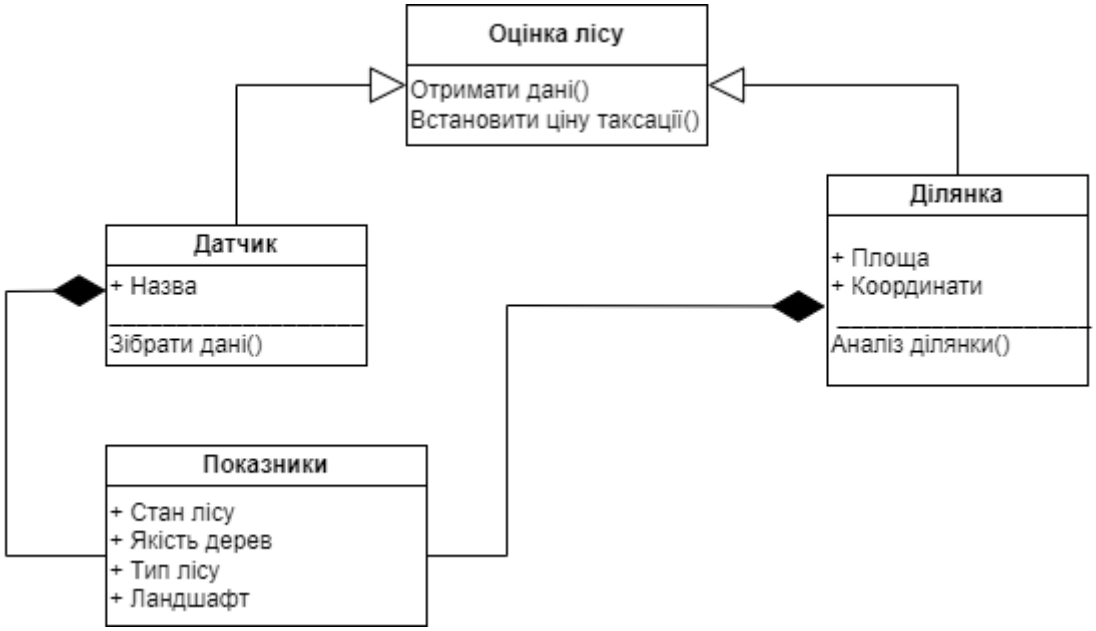

Рис. 8 Діаграма кооперацій

2.3 Діаграма пакетів

Діаграма пакетів UML — це тип структурної діаграми, яка ілюструє організацію та залежності між різними пакетами в системі. Пакет служить контейнером, який групує пов’язані елементи, такі як класи, інтерфейси, компоненти та інші пакети, для полегшення організації та керування.

На діаграмах пакетів пакети представлені у вигляді прямокутників із символом табуляції вгорі, а назва пакета записана всередині прямокутника. Залежності між пакетами зображені стрілками.

Основна мета діаграми пакетів — надати огляд архітектури системи, демонструючи, як різні компоненти та підсистеми організовані в пакети. Це допомагає зрозуміти структуру високого рівня системи, визначити залежності між пакетами та полегшити модульне проектування та розробку.

Діаграми пакетів можна використовувати для візуалізації рівнів або рівнів системи, таких як рівень презентації, рівень бізнес-логіки та рівень доступу до даних у програмному забезпеченні. Вони також можуть представляти залежності між підсистемами, модулями або компонентами в рамках більшої системи[19].

На додаток до відображення структурної організації пакетів, діаграми пакетів можуть містити додаткову інформацію, таку як зв’язки між пакетами (наприклад, узагальнення, реалізація або залежність), вміст пакетів (наприклад, класи або інтерфейси в кожному пакеті) і видимість пакетів (наприклад, загальнодоступних, приватних або захищених).

Діаграми пакетів UML забезпечують високорівневе уявлення про структуру системи, допомагаючи зацікавленим сторонам зрозуміти організацію та залежності між різними пакетами в системі. Вони є цінними інструментами для системних архітекторів, дизайнерів і розробників протягом життєвого циклу розробки програмного забезпечення.

Діаграма пакетів (рис. 9) для цього програмного забезпечення використовується для візуального представлення архітектури та організації системи. У ньому зображено різні пакети, які входять до складу програмного забезпечення, і показано зв’язки та залежності між ними.

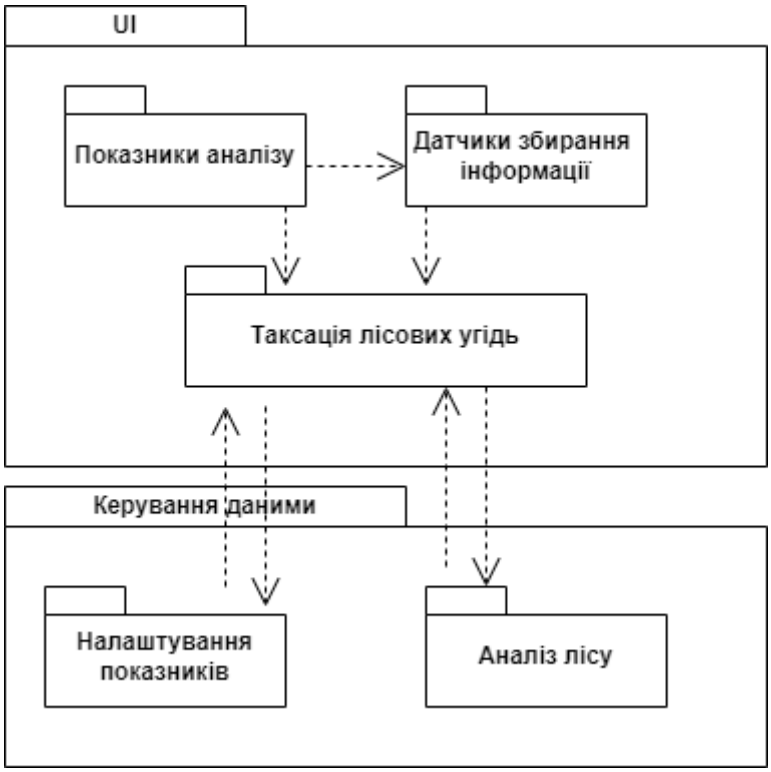

Рис. 9 Діаграма пакетів

Діаграма пакетів ілюструє різні функціональні та структурні компоненти програмного забезпечення, згруповані в пакети. Кожен пакет представляє собою логічне групування пов’язаних класів, інтерфейсів та інших елементів, які сприяють певним функціям або модулям у системі.

На схемі пакети зображені у вигляді прямокутних коробок, усередині яких відображаються їхні назви. Стрілки або лінії використовуються для представлення зв’язків і залежностей між пакетами. Ці зв’язки можуть включати в себе залежності, узагальнення або реалізації, показуючи, як один пакет покладається на інший або розширює його.

2.4 Діаграма компонентів

Діаграма компонентів UML ілюструє організацію та залежності компонентів високого рівня в системі. Він показує, як різні компоненти, такі як бібліотеки, виконувані файли та модулі, взаємодіють, щоб відповідати системним вимогам або надавати певні функції.

Ключові елементи діаграми компонентів включають:

- компоненти: це будівельні блоки системи, що представляють собою модульні одиниці з чітко визначеними інтерфейсами. Вони інкапсулюють функціональність і можуть містити бібліотеки, виконувані файли, файли вихідного коду або інші модулі, які можна розгортати;
- інтерфейси: визначте контракти чи угоди, яких компоненти дотримуються під час взаємодії. Вони визначають методи, операції або послуги, що надаються компонентом, і ті, що вимагаються від інших компонентів;
- залежності: вказують на зв’язки між компонентами. Ці залежності можуть включати залежності використання, коли один компонент використовує послуги, надані іншим, або залежності реалізації, коли компонент реалізує або реалізує інтерфейс, наданий іншим;
- з’єднувачі: представляють канали зв’язку або механізми, за допомогою яких компоненти взаємодіють. Вони можуть включати різні типи протоколів зв’язку, такі як виклики методів, повідомлення або сигнали;
- порти: служать точками взаємодії між компонентами та їх середовищем. Вони визначають інтерфейси, через які компоненти надсилають і отримують повідомлення або дані;
- артефакти: представляють фізичні файли або модулі, створені або розгорнуті під час процесу розробки. Вони можуть містити файли вихідного коду, бібліотеки, виконувані файли, файли конфігурації або будь-який інший матеріальний продукт процесу розробки.

Діаграми компонентів є цінними для розуміння архітектури системи та її модульної структури. Вони допомагають ідентифікувати повторно використовувані компоненти, визначати чіткі інтерфейси та керувати залежностями між різними частинами системи. Крім того, вони полегшують спілкування між зацікавленими сторонами, забезпечуючи візуальне представлення архітектури системи та взаємодії компонентів.

Діаграма компонентів для даної програми представлена на малюнку 10.

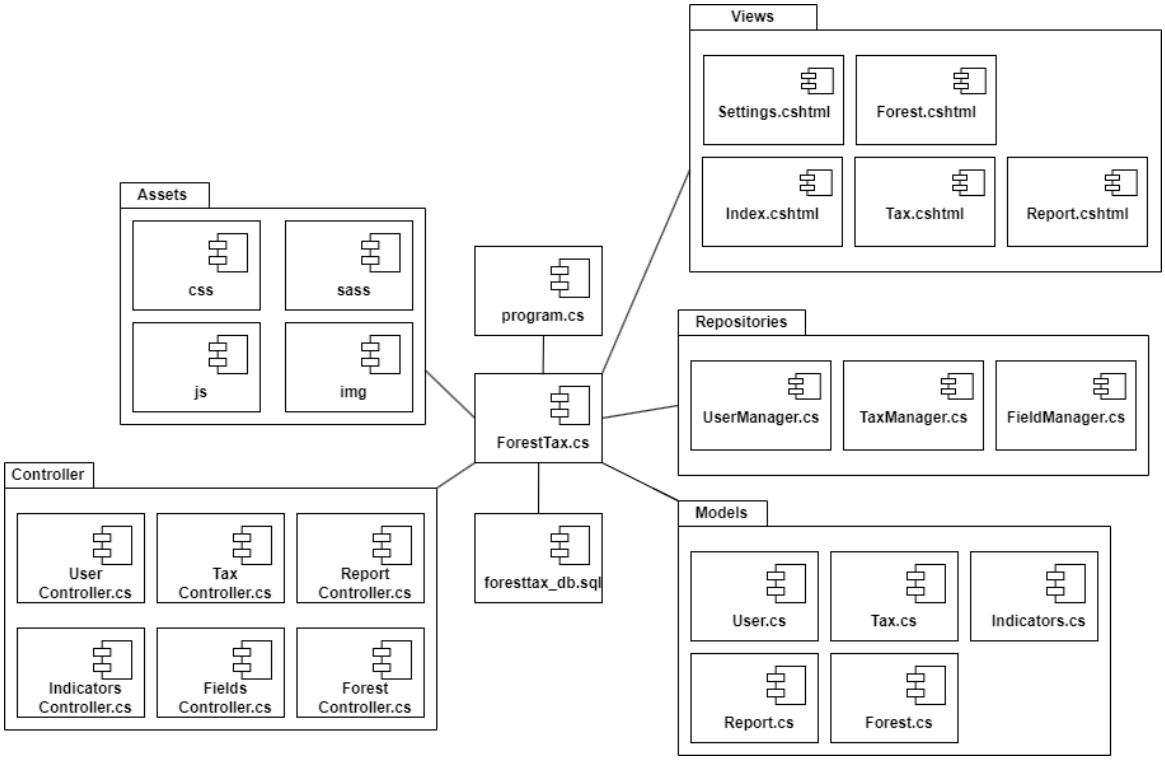

Рис. 10 Діаграма компонентів

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

Реляційна база даних — це тип системи керування базами даних (СУБД), яка організовує та зберігає дані структурованим чином. Він дотримується реляційної моделі, де дані зберігаються в таблицях, що складаються з рядків і стовпців. Кожна таблиця представляє певну сутність, а стовпці представляють атрибути цієї сутності.

У реляційній базі даних таблиці — це первинні структури, що складаються з рядків (записів або кортежів) і стовпців (полів або атрибутів). Рядки представляють окремі екземпляри змодельованих сутностей, тоді як стовпці представляють конкретні характеристики цих сутностей.

Зв’язки встановлюють зв’язки між таблицями, визначаючи, як дані в одній таблиці співвідносяться з даними в іншій. Ці зв’язки базуються на загальних атрибутах або ключах, спільних між таблицями.

Ключі використовуються для унікальної ідентифікації рядків у таблиці. Первинний ключ — це стовпець або комбінація стовпців, які унікально ідентифікують кожен запис. Зовнішні ключі встановлюють зв’язки між таблицями, посилаючись на первинний ключ іншої таблиці.

Нормалізація - це процес організації таблиць для усунення надмірності та підвищення цілісності даних. Це передбачає ефективне структурування даних, зменшення дублювання та забезпечення того, що інформація зберігається лише в одному місці[20].

Структурована мова запитів (SQL) — це стандартизована мова для взаємодії з реляційними базами даних. Це дозволяє користувачам запитувати, змінювати та керувати базою даних. SQL підтримує пошук даних, вставку, оновлення, видалення та визначення схеми бази даних.

Реляційні бази даних пропонують такі переваги, як цілісність даних, гнучкість, масштабованість, підтримка складних операцій і відповідність властивостям ACID (атомарність, послідовність, ізоляція, довговічність). Вони широко використовуються в різних галузях завдяки здатності обробляти великі обсяги даних, забезпечувати незалежність даних і підтримувати надійну обробку транзакцій.

Таким чином, реляційні бази даних забезпечують структурований і ефективний спосіб зберігання та керування даними. Вони використовують таблиці, зв'язки, ключі та SQL для забезпечення цілісності даних, гнучкості, масштабованості та надійної обробки транзакцій. Ці бази даних широко застосовуються в різних програмах і галузях завдяки своїй надійності та здатності ефективно обробляти структуровані дані.

Вибір правильної системи керування базами даних (СУБД) має вирішальне значення для розробки та функціонування системи. Вибір СУБД повинен відповідати вимогам і цілям програмного забезпечення.

Процес відбору розпочався з оцінки конкретних вимог до програмного забезпечення, враховуючи такі фактори, як обсяг даних, робоче навантаження користувача, типи даних, потреби безпеки та масштабованість. Цей аналіз допоміг визначити бажані функції та можливості СУБД.

Було досліджено та оцінено різні параметри систем керування базами даних (СУБД), придатних для програмного забезпечення, включаючи MySQL, PostgreSQL, Oracle, MongoDB або SQL Server. Були враховані такі фактори, як функції, продуктивність, масштабованість, безпека, підтримка спільноти та вартість ліцензування. Було також забезпечено підтримку СУБД необхідних моделей даних і мов запитів[20].

Було оцінено потреби в інтеграції програмного забезпечення для відстеження стану здоров'я та підтримки фізичної активності, щоб визначити, чи може СУБД бездоганно інтегруватися з іншими використовуваними технологіями чи фреймворками. Сумісність і легкість інтеграції мають вирішальне значення для плавного процесу розробки.

Оцінено показники продуктивності опцій СУБД. Такі функції, як індексування, кешування та оптимізація запитів, були визначені для підвищення продуктивності. Варіанти масштабованості, пропоновані СУБД, розглядалися для адаптації до майбутнього зростання та збільшення навантаження на користувачів.

Безпека даних надзвичайно важлива для програмного забезпечення, яке обробляє конфіденційну інформацію користувача. Було перевірено функції безпеки, які надає кожна СУБД, включаючи шифрування, контроль доступу, можливості аудиту та дотримання правил захисту даних.

Розглянуто простоту розробки, адміністрування та обслуговування СУБД. Було знайдено надійну екосистему, документацію та підтримку спільноти, які могли б допомогти вирішити проблеми та надати своєчасну допомогу. Також було оцінено наявність інструментів і фреймворків, що оптимізують операції з базами даних і міграції.

Були розраховані витрати на ліцензування, плата за підтримку та потенційні витрати на інфраструктуру, пов’язані з параметрами СУБД. На підставі оцінки цих факторів було прийнято рішення обрати MS SQL Server для розробки програмного забезпечення для відстеження здоров’я та підтримки фізичної активності.

MS SQL Server розроблено для роботи з великими обсягами даних і забезпечує високу масштабованість, здатність обслуговувати тисячі користувачів, що одночасно працюють, і масштабуватись відповідно до змінних потреб компанії. Крім того, це високопродуктивна база даних, оптимізована для швидкого доступу та пошуку даних. Він підтримує розширені методи індексування та оптимізації запитів, що робить його ідеальним для програм, які потребують швидкої обробки даних.

MS SQL Server має розширені функції безпеки для захисту даних від несанкціонованого доступу, включаючи захист на рівні рядків, завжди зашифровані та прозоре шифрування даних. Крім того, MS SQL розроблено для бездоганної інтеграції з іншими продуктами та технологіями Microsoft,

сприяючи взаємодії з іншими компонентами програмного забезпечення в стеку технологій Microsoft. Він пропонує функції високої доступності, такі як дзеркальне відображення бази даних, автоматичне перемикання після відмови, а також можливості резервного копіювання та відновлення, забезпечуючи безперервний доступ до даних у разі потреби[20].

Після створення логічної моделі була побудована фізична структура реляційної бази даних (РБД) для запропонованої системи. Як згадувалося раніше, MS SQL Server був обраний як система управління базами даних (СУБД) для системи.

3.2 Розробка інформаційної бази

Під час проєктування таблиць у фізичній моделі основою слугували сутності з логічної моделі. Атрибути цих сутностей були використані для розробки структури таблиці. Отриману схему бази даних зображено на рис. 11.

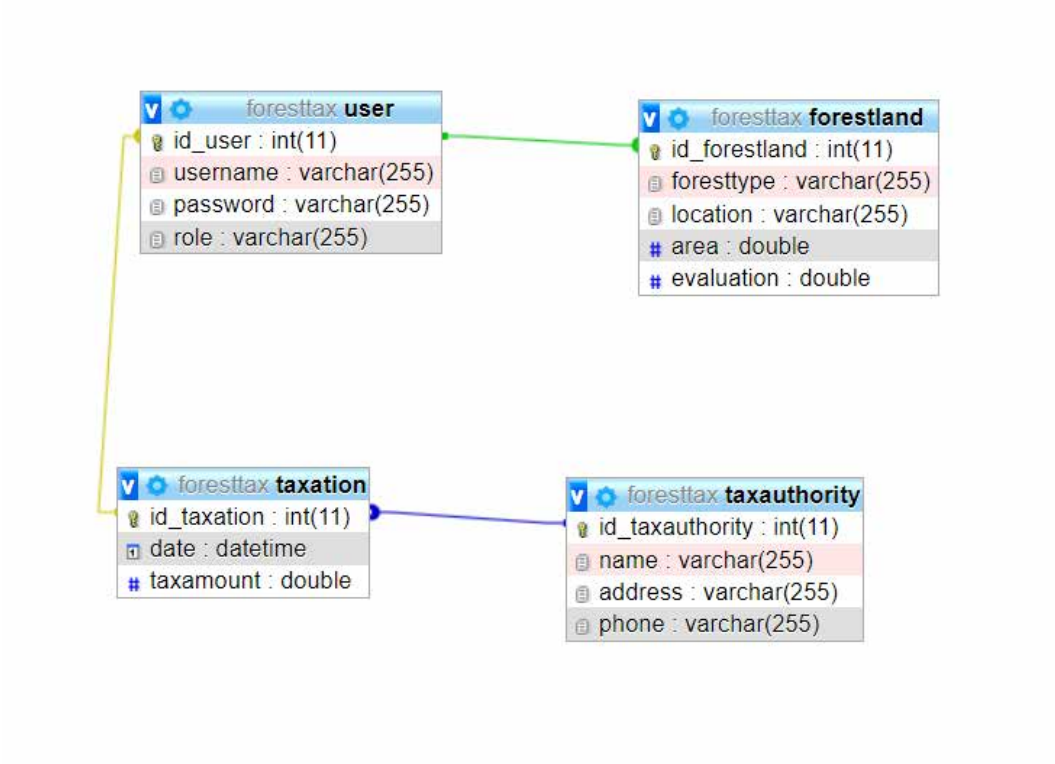

Рис. 11 База даних системи

Розглянемо детальніше створення кожної з таблиць і властивості визначені для них. Опишемо назву поля та тип даних усіх атрибутів.

Таблиця «User», зображена на рис. 12, зберігає в БД користувачів, має такі поля:

- id_user – int(11). Ключове поле таблиці «User»;
- username – varchar(255). Поле, в якому зберігається ім'я користувача;
- password – varchar(255). Поле, в якому зберігається зашифрований пароль користувача;
- role – varchar(255). Поле, в якому визначається роль користувача в системі.

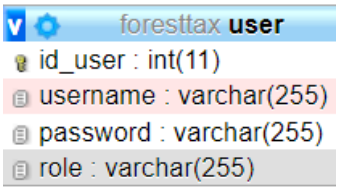

Рис. 12 Створена таблиця «User»

Таблиця «ForestLand», зображена на рис. 13, зберігає в БД дані про лесові ділянки, має такі поля:

- id_forestland – int(11). Ключове поле таблиці «ForestLand»;
- foresttype – varchar(255). Поле, в якому зберігається тип лісу;
- location – varchar(255). Поле, в якому зберігається місцезнаходження ділянки;
- area – double. Поле, в якому визначається площа ділянки;
- Evaluation – double. Поле, в якому зберігається примірна вартість ділянки.

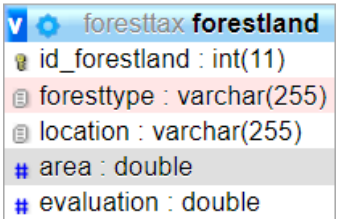

Рис. 13 Створена таблиця «ForestLand»

Таблиця «Taxation», зображена на рис. 14, зберігає в БД дані про таксацію лісових угідь, має такі поля:

- id_taxation – int(11). Ключове поле таблиці «Taxation»;
- date – datetime. Поле, в якому зберігається дата оцінки;
- taxamount – double. Поле, в якому визначається ціна лісових угідь.

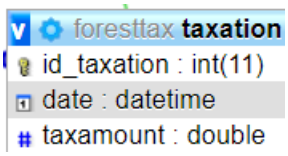

Рис. 14 Створена таблиця «Taxation»

Таблиця «TaxAuthority», зображена на рис. 15, зберігає в БД дані про власників лісових угідь, має такі поля:

- id_taxauthority – int(11). Ключове поле таблиці «TaxAuthority»;
- name – varchar(255). Поле, в якому зберігається ім’я власника;
- address – varchar(255). Поле, в якому зберігається адреса власника;
- phone – varchar(255). Поле, в якому зберігається телефон власника.

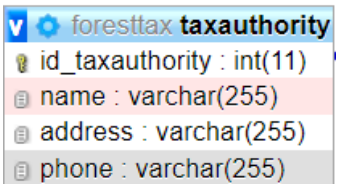

Рис. 15 Створена таблиця «TaxAuthority»

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Інструментарій розробки автоматизованої системи таксації земель лісового фонду в польових умовах включає низку потужних засобів і технологій.

В основі стеку розробки лежить C#, універсальна та об’єктно-орієнтована мова програмування, яка використовується для реалізації логіки

та функцій серверної частини. Доповненням до C# є ASP.NET, надійна структура, що полегшує створення динамічних та інтерактивних веб-сторінок.

EF Core (Entity Framework Core) спрощує взаємодію з базою даних, пропонуючи високорівневу абстракцію над базою даних, зменшуючи потребу в необроблених запитах SQL. Ідентифікація, інший невід'ємний компонент, забезпечує безпечну автентифікацію користувача, авторизацію та керування в програмі.

Razor Pages, модель програмування на основі сторінок, оптимізує розробку динамічних веб-сторінок шляхом поєднання коду HTML і C#.

Для зберігання та керування даними використовується MS SQL Server як система керування реляційною базою даних, що забезпечує масштабованість і розширені функції безпеки.

У інтерфейсі HTML, CSS і JavaScript використовуються для створення інтерфейсу користувача, а Bootstrap надає попередньо розроблені компоненти інтерфейсу користувача та таблиці стилів для покращеної візуальної привабливості та оперативності.

Використовуючи цей комплексний інструментарій, розробники можуть створювати складне та ефективне програмне рішення для автоматизації процесів оподаткування лісових земель, забезпечуючи точність і зручність використання навіть у польових умовах.

3.4 Архітектура програмного забезпечення

У контексті розробки програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах варто розглянути кілька прикладних архітектур, кожна зі своїми характеристиками та придатністю.

Монолітна архітектура передбачає побудову всієї системи як єдиного, тісно інтегрованого блоку. Усі компоненти, включаючи інтерфейс

користувача, бізнес-логіку та рівень доступу до даних, об’єднані в єдину кодову базу. Хоча монолітні архітектури спочатку легко розробити та розгорнути, вони можуть стати складнішими та складними в обслуговуванні в міру зростання системи. Однак вони можуть бути придатними для невеликих програм з обмеженою функціональністю.

Сервісно-орієнтована архітектура розбиває систему на модульні сервіси, які можна розгортати незалежно один від одного, які взаємодіють між собою через стандартизовані інтерфейси. Кожна служба зосереджена на певних функціях, таких як збір даних, розрахунок податків або звітність. SOA сприяє багаторазовому використанню, масштабованості та гнучкості, що робить його придатним для систем із різноманітними вимогами, що постійно змінюються, як-от програми оподаткування лісових земель.

Подібно до SOA, архітектура мікросервісів розкладає систему на невеликі автономні служби. Однак мікросервіси, як правило, мають менший обсяг і часто побудовані навколо конкретних бізнес-можливостей. Кожен мікросервіс відповідає за одну функцію та взаємодіє з іншими сервісами через спрощені протоколи, такі як HTTP або черги обміну повідомленнями. Архітектура мікросервісів забезпечує більшу гнучкість, масштабованість і стійкість, але може ускладнити керування розподіленими системами.

EDA наголошує на виробництві, виявленні, споживанні та реакції на події, що відбуваються в системі. Події генеруються різними компонентами та можуть ініціювати дії в інших частинах системи. EDA є корисним для систем, де обробка даних у реальному часі, асинхронний зв’язок і реагування на зміни умов є критичними, що добре узгоджується з динамічною природою польових умов у оподаткуванні лісових земель.

Рівнева архітектура розділяє систему на окремі рівні, такі як презентація, бізнес-логіка та доступ до даних, причому кожен рівень має певні обов’язки. Ця архітектура сприяє модульності, масштабованості та ремонтопридатності завдяки встановленню чітких меж між компонентами. Хоча багаторівнева архітектура може бути не такою гнучкою, як SOA або мікросервіси, вона може

бути придатною для програм із відносно стабільними вимогами та простішими моделями взаємодії.

Серед цих архітектур вибір для розробки програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах, швидше за все, схилятиметься до сервісно-орієнтованої архітектури (SOA) або архітектури мікросервісів через їх модульність, масштабованість, гнучкість і здатність адаптувати різноманітні та розвиваються вимоги, а також враховуючи такі фактори, як розмір системи, складність і очікуване зростання.

Вибір сервіс-орієнтованої архітектури (SOA) для розробки програмного забезпечення для автоматизованої системи оподаткування земель лісового фонду в польових умовах має кілька вагомих причин.

По-перше, SOA дозволяє розбивати систему на менші незалежні служби, кожна з яких зосереджується на конкретних аспектах оподаткування лісових земель, таких як збір даних, розрахунок податків або звітність. Цей модульний підхід полегшує масштабованість, дозволяючи додавати або змінювати служби за потреби.

По-друге, SOA сприяє гнучкості та багаторазовому використанню, створюючи слабко пов’язані служби. Ця гнучкість дозволяє повторно використовувати служби в різних частинах системи або навіть в інших програмах, тим самим підвищуючи ефективність і скорочуючи час розробки.

По-третє, SOA сприяє бездоганній інтеграції із зовнішніми системами або базами даних, такими як географічні інформаційні системи (ГІС) або урядові бази даних. Стандартизовані інтерфейси та протоколи забезпечують взаємодію з існуючими системами.

Крім того, SOA підтримує розподілене середовище, що робить його придатним для польових умов, де підключення може бути обмеженим. Сервіси можна розгортати в розподілених середовищах, забезпечуючи ефективну роботу системи навіть у складних умовах.

Крім того, SOA наголошує на можливості виявлення сервісів, полегшуючи пошук компонентів системи та динамічну взаємодію один з

одним. Ця функція особливо корисна в динамічних польових умовах, коли компоненти можуть непередбачувано вмикатися або виходити з мережі.

Крім того, децентралізована природа SOA забезпечує ізоляцію помилок, сприяючи стійкості та надійності. Якщо одна служба виходить з ладу, це не обов'язково вплине на всю систему.

Нарешті, SOA забезпечує горизонтальну масштабованість, дозволяючи розгортати додаткові екземпляри служб для обробки збільшеного робочого навантаження, що має вирішальне значення для систем, які відчувають коливання попиту.

Таким чином, впровадження SOA для розробки програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах пропонує такі переваги, як модульність, масштабованість, гнучкість, інтеграція, відмовостійкість і стійкість, що робить його відповідним архітектурним підходом для вирішення унікальних завдань таких систем.

Нижче ми продемонструємо SOA архітектуру цього програмного забезпечення (рис. 16).

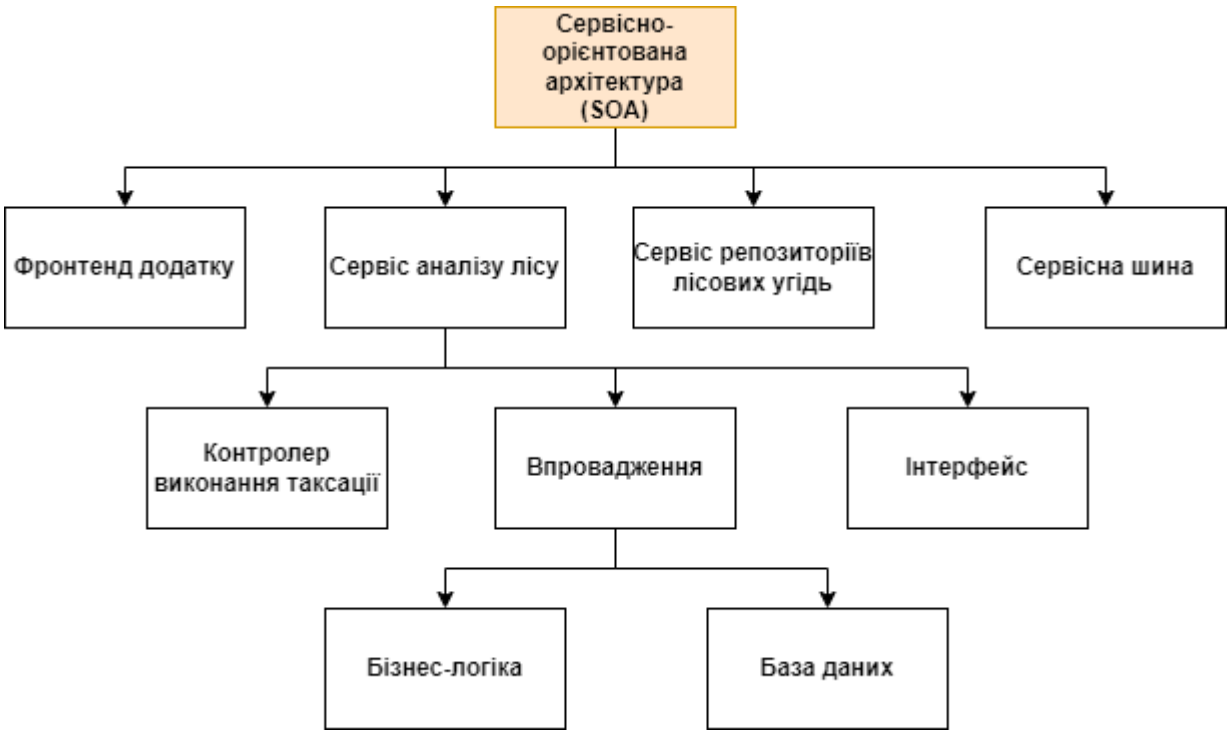

Рис. 16 SOA архітектура програмного забезпечення

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування програмного забезпечення – це процес оцінювання якості програмного забезпечення для виявлення дефектів, помилок або не відповідностей. Його основна мета — забезпечити, щоб програмне забезпечення працювало належним чином, виконувало необхідні функції та відповідало вимогам користувача.

Процес тестування передбачає створення сценаріїв тестування, які визначають дії та вхідні дані для перевірки різних аспектів програми. Тестування може бути автоматизованим або виконаним вручну, кожен підхід має свої переваги та застосування залежно від контексту проекту та вимог.

Основні цілі тестування програмного забезпечення включають:

- виявлення дефектів: тестування допомагає виявити помилки, недоліки та дефекти програмного забезпечення, дозволяючи їх виправити до випуску продукту;
- підтвердження відповідності вимогам: тестування перевіряє, чи відповідає програмне забезпечення встановленим вимогам, включаючи функціональні та нефункціональні вимоги;
- забезпечення якості: тестування забезпечує високу якість програмного забезпечення, перевіряючи його функціональність, стабільність, надійність і продуктивність;
- зміцнення впевненості користувачів: тестування дає користувачам впевненість у функціональності та якості програмного забезпечення, зменшуючи ризик непередбачених проблем;
- покращення взаємодії з користувачем: тестування допомагає виявити та вирішити проблеми, які можуть вплинути на зручність

використання програмного забезпечення та задоволення користувачів.

Тестування системи проводилося на основі тестування інтерфейсу користувача та функціонального тестування.

Запустивши систему бачимо сторінку авторизації програмного забезпечення (рис. 16).

Рис. 16 Сторінка авторизації програмного забезпечення

При введенні некоректних даних у поля авторизації отримуємо повідомлення про необхідність відкоригувати дані (рис.17).

Рис. 17 Сторінка авторизації з повідомленнями про помилки

Якщо ж такий користувач існує, та дані авторизації були вірні, то користувача переносить до головного вікна системи (рис.18).

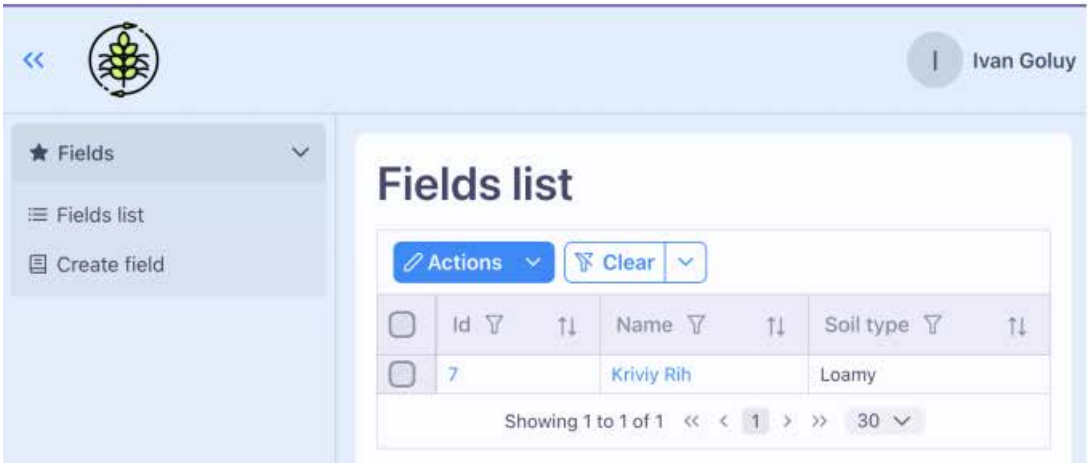

Рис. 18 Головна сторінка системи з початковими даними списку доступних угідь

Залежно від типу користувача, а користувачем у даному випадку виступає професіонал лісового господарства, будуть відображені різні можливості, системні адміністратори будуть лише можливості переглядати доступні їм поля, та виконувати з ними маніпуляції. Професіоналам лісового господарства буде доступний додатковий функціонал у вигляді управління користувачами, лесовими угіддями, датчиками (Рис. 19)

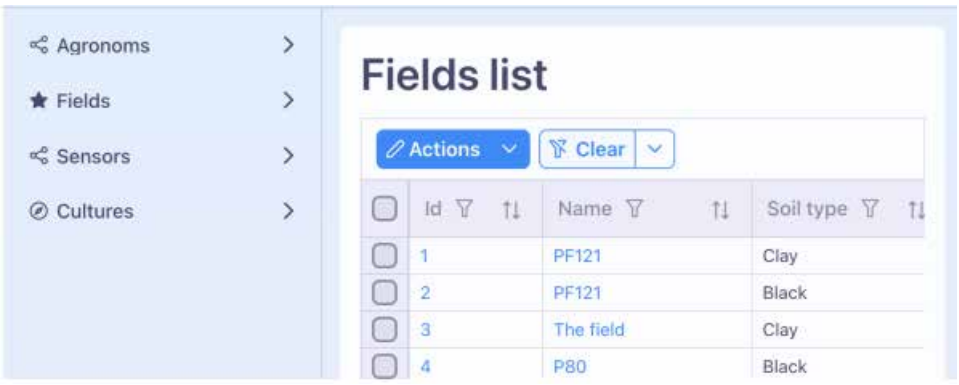

Рис. 19 Головна сторінка системи для професіонала лісового господарства

Далі, як і у професіонала, так і у адміна є можливості перегляду лісових угідь, для яких потрібно вказати розташування за допомогою географічних координат (Рис. 20)

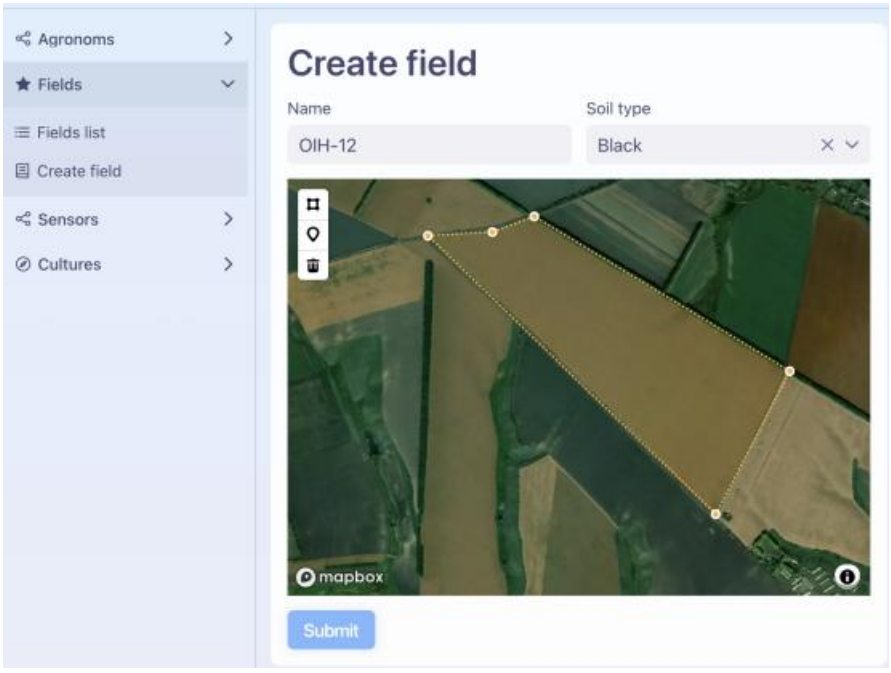

Рис. 20 Сторінка перегляду угіддя

Після створення угіддя є можливість перегляду даних угіддя, де буде зображено сам ліс, та пов’язані з ним датчики, якщо такі на ньому присутні (Рис. 21)

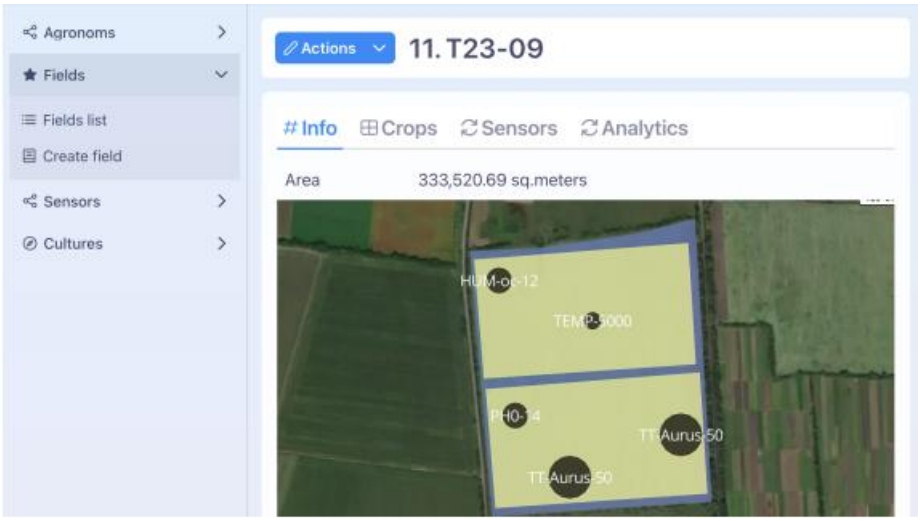

Рис. 21 Сторінка перегляду інформації по угіддю

4.2 Вимоги до апаратного та програмного забезпечення

Вимоги до апаратного забезпечення комп’ютера для запуску програмного забезпечення: процесор із принаймні двома ядрами, наприклад

Intel Core i3 або AMD Ryzen 3; 4 Гб оперативної пам'яті; SSD з ємністю 10 ГБ або більше.

Діаграма розгортання ілюструє загальну топологію розподіленої системи, показуючи, як різні артефакти представлені на окремих вузлах системи. Його мета — візуалізувати фізичну структуру системи, зображуючи, як програмні компоненти розгортаються на різних вузлах.

Вузли на діаграмі зображені у вигляді прямокутників або піктограм, що представляють апаратні пристрої або середовища виконання, на яких працює система. Програмні артефакти, такі як програми, модулі чи служби, представлені у вигляді прямокутників або піктограм, розміщених у вузлах, щоб вказати їхнє розгортання.

Використовуючи зв’язки розгортання, такі як асоціації, залежності або залежності розгортання, діаграма демонструє зв’язки та взаємодію між компонентами програмного забезпечення та апаратними вузлами. Ці зв’язки допомагають зрозуміти, як компоненти спілкуються та співпрацюють у системній інфраструктурі.

Серверна сторона використовується для обробки запитів, взаємодії з базою даних, зв’язку з іншими серверами, обробки введених користувачем даних і структурування веб-додатків. Клієнт — це користувач, який взаємодіє з програмою за допомогою веб-браузера.

Сервер бази даних — це програмний ресурс, який використовується для зберігання даних у системі.

На малюнку 22 зображено схему розгортання цієї системи. Архітектура клієнт-сервер реалізована на двох окремих серверах.

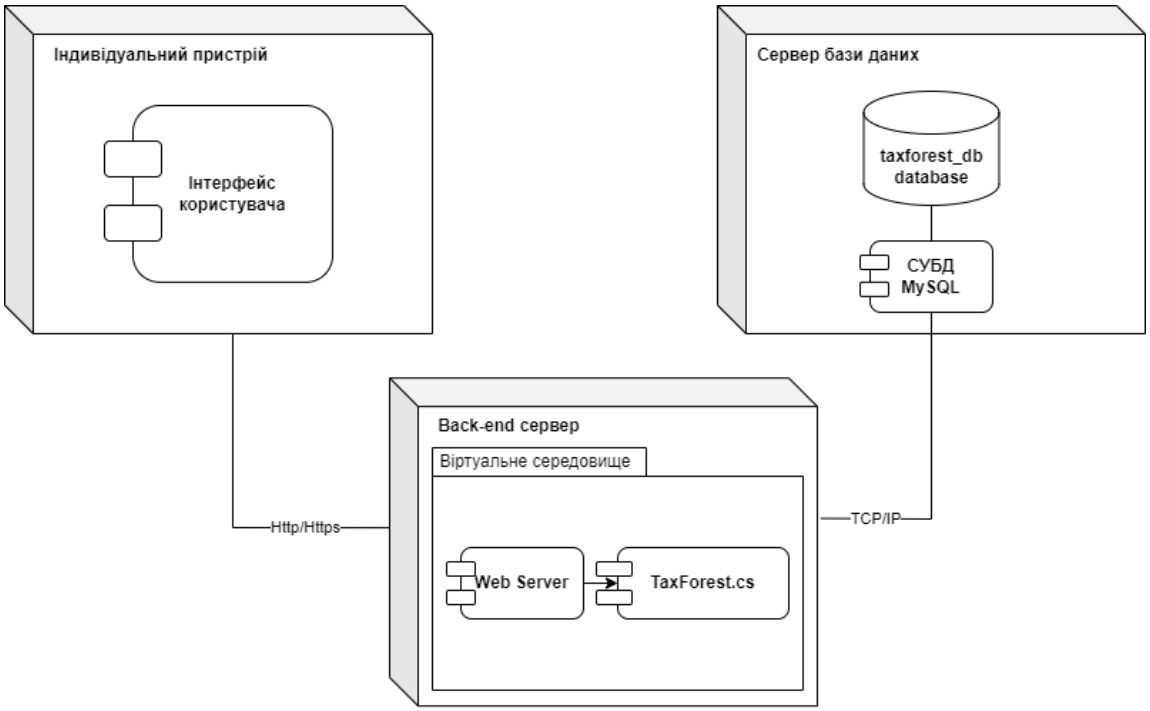

Рис. 22 Діаграма розгортання

Апаратні вимоги до системи, де буде знаходитись бекенд додаток можуть мати такий вигляд:

− процесор (CPU): рекомендується мати потужний багатоядерний процесор, оскільки бекенд сервер виконує обчислювальну інтенсивні операції. Мінімум може бути чотирьох-ядерний процесор, але для більшої продуктивності краще мати вісім або більше ядер;

− оперативна пам'ять (RAM): мінімальний обсяг оперативної пам'яті залежить від розміру додатка та очікуваного навантаження. Рекомендується мати не менше 8 ГБ оперативної пам'яті. Для більших і складних проектів може знадобитись 16 ГБ або більше;

− дисковий простір: мінімально рекомендований обсяг дискового простору становить 20-30 ГБ, проте врахуйте, що розмір може збільшуватись з часом.

ВИСНОВКИ

Підсумовуючи, бакалаврська кваліфікаційна робота спрямована на розробку інформаційної системи обліку та оподаткування земель лісового фонду в польових умовах є значним прогресом у сфері управління та оподаткування лісового господарства. Завдяки використанню сучасних технологій і принципів програмної інженерії така система може впорядкувати й оптимізувати процес оцінки та оподаткування лісових земель, що призведе до більш ефективних і точних результатів.

Робота розпочалася з вивчення предметної області та формулювання проблеми дослідження. Існуючі рішення були переглянуті, щоб виявити прогалини та можливості для вдосконалення програмного забезпечення для оподаткування земель. Це послужило основою для наступних етапів роботи.

Включаючи такі функції, як географічні інформаційні системи (GIS), аналітика даних і автоматизація, програмне забезпечення може підвищити продуктивність професіоналів лісового господарства, одночасно забезпечуючи дотримання податкових норм і екологічних стандартів. Крім того, система може полегшити процеси прийняття рішень на основі даних, дозволяючи зацікавленим сторонам робити обґрунтований вибір щодо управління лісами та зусиль щодо збереження.

Створено логічну модель даних, що представляє вимоги до інформації та зв’язки в програмному забезпеченні для таксації лісових угідь. Ця модель лягла в основу проектування програмної системи. Розроблено фізичну модель даних, що забезпечує ефективне зберігання та пошук даних за допомогою вибраної системи керування базами даних.

Архітектура програмного забезпечення ретельно розроблена з урахуванням вимог системи та її масштабованості. Вибір відповідних інструментів розробки, таких як ASP.NET Core 6 MVC, C#, MS SQL Server,

Bootstrap 5, HTML, CSS і JS, був зроблений для забезпечення надійності, функціональності та зручності для користувача.

Етап впровадження включав переведення дизайну в фактичний код та інтеграцію різних програмних компонентів. Для забезпечення функціональності та надійності програмного забезпечення було проведено ретельне тестування та налагодження. Відгуки користувачів і повторювані вдосконалення відіграли важливу роль у підвищенні продуктивності та зручності використання програмного забезпечення.

Крім того, процес розробки такого програмного забезпечення вимагає ретельного розгляду вимог користувачів, відгуків зацікавлених сторін і технічної здійсненності. Він включає різні етапи, включаючи аналіз вимог, проектування, впровадження, тестування та розгортання. Крім того, постійне технічне обслуговування та підтримка необхідні для забезпечення надійності та ефективності системи з часом.

Таким чином, розробка програмного забезпечення для автоматизованої системи оподаткування лісових земель у польових умовах має величезний потенціал для революції в лісовій галузі, надаючи комплексне та ефективне рішення для оцінки та оподаткування землі. Завдяки співпраці між розробниками програмного забезпечення, експертами з лісового господарства та державними установами ця технологія може сприяти сталим методам управління лісами та зусиллям із збереження навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Forest Inventory and Analysis Database: Database description and users manual [Електронний ресурс] - Режим доступу: https://www.fs.usda.gov/rds/archive/Product/RDS-2017-0006/

2. Методи та практика оцінки лісової таксі в польових умовах: Огляд літератури – Режим доступу: https://doi.org/10.1016/j.foreco.2019.117531

3. Застосування геоінформаційних систем (ГІС) у лісовому господарстві: огляд – Режим доступу: https://doi.org/10.3390/f8120462

4. Дистанційне зондування та ГІС для аналізу та управління лісовими екосистемами - Режим доступу: https://www.springer.com/gp/book/9783540779049

5. Інформаційні системи управління лісами: огляд – Режим доступу: https://doi.org/10.1007/s00267-020-01357-2

6. Automation in Forestry: Real-World Solutions - Режим доступу: https://www.springer.com/gp/book/9783030696701

7. Довідник з лісового оподаткування: посібник для професіоналів лісового господарства - Режим доступу: https://www.cabi.org/bookshop/book/9781845931762

8. Вступ до геоінформаційних систем - Режим доступу: https://www.esri.com/en-us/what-is-gis/history-of-gis

9. Лісове застосування повітряного лазерного сканування - Режим доступу: https://www.taylorfrancis.com/books/9780429126563

10. Дистанційне зондування та інтерпретація зображень - Режим доступу: https://www.wiley.com/en-us/Remote+Sensing+and+Image+Interpretation%2C+7th+Edition-p-9781118343289

11. Закони та правила оподаткування лісів: порівняльний аналіз - Режим доступу: https://link.springer.com/article/10.1007/s10531-020-01957-8

12. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ru/glossary/mnogourovnevaya-arhitektura/

13. Багаторівнева архітектура в ASP.NET MVC – [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/mvc5/23.5.php

14. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724

15. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

16. ПРОЕКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/

17. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html

18. ПРОЕКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

19. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

20. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

21. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html

22. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ru/uml-classes/

23. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html

24. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html

25. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.

26. Джонсон, А. (2021). «Інформаційні системи управління лісами: комплексний огляд». Міжнародний журнал досліджень лісового господарства, 2021, 1-15.

27. Лі К. К. (2019). «Застосування дистанційного зондування в лісовій інвентаризації та оподаткуванні: аналіз прикладів». Журнал прикладного дистанційного зондування, 13(4), 042704.

28. Wang, Q., & Xu, C. (2020). «Геоінформаційні системи (ГІС) у таксації лісів: сучасні тенденції та майбутні напрями». Міжнародний журнал цифрової Землі, 13(9), 1018-1035.

29. Патель, Р., і Шарма, С. (2018). «Автоматизація в лісовому господарстві: виклики та можливості». Журнал автоматизації в екології, 15 (2), 98-110.

30. Браун, Т. Р. (2022). «Економічний аналіз політики оподаткування лісів: порівняльне дослідження». Журнал економіки навколишнього середовища та менеджменту, 105, 102433.

ДОДАТОК А

Фрагменти програмного коду. Функція аналізу лісової ділянки

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;

namespace ForestTaxationSystem.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ForestAnalysisController : ControllerBase
    {
        // POST api/forestanalysis
        [HttpPost]
        public IActionResult Post([FromBody] List<Tree> trees)
        {
            if (trees == null || trees.Count == 0)
            {
                return BadRequest("No tree data provided.");
            }

            int totalTrees = trees.Count;
            int totalCarbon = 0;
            int totalVolume = 0;

            foreach (var tree in trees)
            {
                int carbonSequestration = CalculateCarbonSequestration(tree);
                totalCarbon += carbonSequestration;

                totalVolume += tree.Volume;
            }

            var analysisResults = new Dictionary<string, int>
            {
                { "TotalNumberOfTrees", totalTrees },
                { "TotalCarbonSequestered", totalCarbon },
                { "TotalVolumeOfTrees", totalVolume }
            };

            return Ok(analysisResults);
        }
```

```
        private int CalculateCarbonSequestration(Tree tree)
        {
            return tree.Volume * 10;
        }
    }

    public class Tree
    {
        public int Volume { get; set; }
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Функція таксації лісової ділянки

```
using System;
using System.Collections.Generic;
```

```
using Microsoft.AspNetCore.Mvc;

namespace ForestTaxation.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ForestTaxController : ControllerBase
    {
        // POST api/foresttax
        [HttpPost]
        public  ActionResult<double>  CalculateForestTax([FromBody]  List<Tree>
trees)
        {
            if (trees == null || trees.Count == 0)
            {
                return BadRequest("No trees provided for taxation.");
            }

            double totalTax = 0;
            foreach (var tree in trees)
            {
                double treeTax = CalculateTreeTax(tree);
                totalTax += treeTax;
            }

            return Ok(totalTax);
        }

        private double CalculateTreeTax(Tree tree)
        {
            double tax = 0;

            switch (tree.Species)
            {
                case "Oak":
                    tax = (tree.Diameter * 1.2) / 2;
                    break;
                case "Pine":
                    tax = (tree.Diameter * 0.8) / 2;
                    break;
                case "Maple":
                    tax = (tree.Diameter * 1.0) / 2;
                    break;
```

```
                default:
                    tax = (tree.Diameter * 0.5) / 2;
                    break;
            }

            return tax;
        }
    }

    public class Tree
    {
        public string Species { get; set; }
        public double Diameter { get; set; }
    }
}
```