НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ Ігор БОЛБОТ	**_____ Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення системи відслідковування динаміки розвитку сільськогосподарських культур»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Георгій БОРОДКІН**
науковий ступінь та вчене звання (підпис)

Виконав _____ **Дмитро БОРТНІКОВ**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

« __ » ____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Бортнікову Дмитру Сергійовичу

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення системи відслідковування динаміки розвитку сільськогосподарських культур»
затверджена наказом від « 19 » грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру « 22 » травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Відомості про моделювання агротехнічних та економічних процесів у рослинництві, організацію взаємодії користувача з базою даних для розрахунку вегетаційних індексів і формування фінансової звітності.
Перелік питань, що підлягають дослідженню:

- Аналіз предметної області та порівняльне дослідження існуючих систем агроменеджменту і симуляції сільськогосподарських процесів.
- Проєктування архітектури бази даних та графічного інтерфейсу управління полями й агроресурсами.
- Програмна реалізація математичних моделей росту культур, бізнес-логіки агротехнічних операцій та калькуляції фінансових показників.
- Тестування функціональних можливостей системи та організація взаємодії програмних компонентів із реляційною базою даних.

Дата видачі завдання « 06 » січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ **(підпис)** **Георгій БОРОДКІН**

Завдання взяв до виконання _____ **(підпис)** **Дмитро БОРТНІКОВ**

ЗМІСТ

ВСТУП

Актуальність завдання, що вирішується. Сучасний етап розвитку світового та вітчизняного агропромислового комплексу характеризується стрімкою цифровізацією та переходом до концепції точного землеробства. Ефективне управління сучасним рослинництвом вимагає безперервного моніторингу значної кількості параметрів навколишнього середовища, стану ґрунту та біометричних показників самих культур. Традиційні методи візуального оцінювання посівів агрономами вже не спроможні забезпечити необхідну швидкість та точність прийняття рішень в умовах нестабільних кліматичних змін, що спостерігаються в останні роки.

Аналіз науково-прикладних джерел свідчить, що автоматизація збору даних та моделювання біологічних процесів є ключовими факторами підвищення врожайності та мінімізації ризиків. Існуючі комерційні програмні комплекси та хмарні сервіси (наприклад, Cropio, OneSoil, Climate FieldView) володіють потужним інструментарієм для глобального супутникового моніторингу та побудови карт вегетаційних індексів (NDVI). Проте, поточний стан науково-прикладного завдання має низку невирішених питань, серед яких:

- висока вартість ліцензій та складна інтеграція існуючих платформ у бізнес-процеси малих та середніх фермерських господарств;
- слабка пов'язаність супутникових або кліматичних даних із локальним мікроекономічним обліком витрат безпосередньо в момент проведення агрозаходу;
- відсутність гнучких локальних інструментів для симуляції погодних сценаріїв (температурних коливань, аномальних опадів) на рівні конкретної секції поля без підключення до дорогих закритих API.

Таким чином, виникає науково-практичне протиріччя між необхідністю точного локального моделювання посівів і доступністю відповідного програмного забезпечення для кінцевого споживача. Місце даної дипломної

роботи у вирішенні цієї задачі полягає у розробці архітектурно гнучкого, локального програмного забезпечення («FieldSimulator»), яке інтегрує математичне моделювання екологічних факторів таких як водний баланс, споживання макроелементів NPK, динаміка росту біомаси . Усе це йде з покроковим фінансовим калькулятором витрат підприємства.

Мета розробки програмного додатку. Метою дипломного проєкту є проектування та програмна реалізація інформаційної системи моніторингу та прогнозування динаміки розвитку сільськогосподарських культур, яка забезпечує автоматизований збір і аналіз щотижневих параметрів посівів, моделювання їхнього стану під впливом кліматичних і антропогенних чинників, а також розрахунок фінансової ефективності проведених агротехнічних заходів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести системний аналіз предметної області та сформувати вимоги до архітектури системи.
2. Побудувати концептуальні, об'єктні та реляційні моделі даних для представлення структури полів і динаміки вимірювань.
3. Розробити математичну модель щотижневої симуляції стану культури (включаючи індекси вологозабезпеченості, азотного балансу та агро-економічної ефективності).
4. Реалізувати багатошарову програмну архітектуру із забезпеченням транзакційної цілісності даних на рівні СКБД.
5. Провести тестування розробленого програмного продукту та оцінити його функціональні характеристики.

Методи та технології, які використовуються при розробці. Для вирішення поставлених завдань було використано комплекс методів системного аналізу, теорії проектування інформаційних систем та інженерії програмного забезпечення. На етапі аналізу предметної області та моделювання використано об'єктно-орієнтований аналіз та мову уніфікованого моделювання UML (діаграми прецедентів, класів, пакетів, компонентів, розгортання та кооперації).

Програмна реалізація додатку здійснена з використанням об'єктно-орієнтованої мови програмування високого рівня C# на базі платформи .NET Framework (версії 4.7.2). Для розробки графічного інтерфейсу користувача (UI Layer) застосовано технологію Windows Forms із інтеграцією сучасної бібліотеки компонентів Flat UI - MetroFramework. Як система керування базами даних (СКБД) обрано реляційну модель Microsoft SQL Server, взаємодія з якою реалізована за допомогою низькорівневої технології ADO.NET (об'єкти SqlConnection, SqlCommand, SqlTransaction) для досягнення максимальної швидкодії та повного контролю над транзакціями.

Апробація розробки програмного додатку.

Апробація розробки відбулась на Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених "Інформаційні технології: економіка, техніка, освіта" (травень 2026 року, м. Київ, НУБІП) та на Міжнародній науково-практичній конференції студентів і молодих учених «Інформаційні технології в соціокультурній сфері, освіті та економіці (травень 2026 року, м. Київ, КНУКІМ).

Структура пояснювальної записки. Пояснювальна записка дипломного проєкту складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 64 сторінки друкованого тексту. Робота містить 22 рисунка. Список використаних джерел включає 22 найменування.

Короткий огляд розділів роботи:

- **У Розділі 1** «Системний аналіз предметної області» наведено детальний опис процесів моніторингу посівів, проведено аналіз функціональних та нефункціональних вимог до системи з погляду власника та користувача, виконано UML-моделювання прецедентів, здійснено порівняльний аналіз існуючих аналогів та сформовано чітке технічне завдання.
- **У Розділі 2** «Проєктування архітектури та математичного забезпечення системи» обґрунтовано вибір трирівневої архітектури, представлено

діаграми класів та пакетів, розроблено структуру реляційної бази даних (3НФ) та детально описано фізичну топологію розміщення компонентів.

- **У Розділі 3** «Розробка інформаційного та програмного забезпечення» описано особливості кодування клієнтської частини додатка на C#, наведено алгоритми взаємодії з СКБД MS SQL Server за допомогою транзакцій, а також описано науково обґрунтований математичний апарат, що лежить в основі симуляційного модуля (водний баланс, GDD, Закон Лібіха).
- **У Розділі 4** «Рекомендації щодо впровадження та експлуатації системи» представлено результати функціонального тестування системи за методом «чорного ящика», сформовано вимоги до апаратного та системного забезпечення, а також визначено структуру інсталяційного пакету з використанням утиліти Inno Setup.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметною областю даного дослідження є процеси оперативного моніторингу, моделювання та фінансово-економічного обліку в сучасному рослинництві в межах концепції точного землеробства. Головна увага приділяється автоматизації контролю життєдіяльності сільськогосподарських культур на локальних ділянках протягом вегетаційного періоду.

Організаційна та територіальна структура сучасного агропідприємства має чітку ієрархію. Базовою великомасштабною одиницею є поле, яке характеризується загальною площею, географічним розташуванням та типом ґрунту. Проте, з точки зору технологічних процесів, поле часто є неоднорідним. Воно може поділятися на окремі ділянки - секції. Секція поля виступає основним неподільним об'єктом управління та моніторингу в інформаційній системі. Саме на рівні секції приймаються агрономічні рішення, оскільки на ній вирощується конкретний вид сільськогосподарської культури, дотримуються однакові терміни посіву, норми внесення добрив та графіки поливу.

Процес моніторингу стану посівів є циклічним і зазвичай прив'язаний до щотижневого інтервалу часу. Протягом цього періоду на кожній секції відбувається безперервна взаємодія трьох основних груп чинників:

1. **Кліматичні (зовнішні екологічні) чинники.** До них належать середньодобова температура повітря та кількість природних опадів. Температура є критичним регулятором біологічних процесів: існує так зване «температурне вікно» активної вегетації (переважно від 10°C до 30°C), всередині якого рослина здатна нарощувати біомасу та ефективно поглинати поживні речовини. Вихід температури за межі цього діапазону уповільнює або повністю зупиняє ріст. Кількість опадів безпосередньо впливає на гідрологічний режим ґрунту.

2. **Показники стану середовища (ґрунту) та біометрія рослин.** Стан ґрунту описується його поточною вологістю (у відсотковому еквіваленті від повної вологоємності) та концентрацією основних макроелементів, необхідних для живлення рослин - азоту (N), фосфору (P) та калію (K). Біометричний стан самої культури на кожному етапі оцінюється через показник її висоти (або об'єму накопиченої біомаси). Усі ці параметри є динамічними: рослина в процесі росту споживає макроелементи та вологість, змінюючи хіміко-фізичний склад ґрунту. Швидкість цього споживання залежить від поточної фази розвитку та температури довкілля. Окрім того, волога природним чином випаровується з ґрунту, і швидкість евапотранспірації нелінійно зростає підвищенням температури.
3. **Антропогенні (агротехнічні) чинники.** Це активні цілеспрямовані втручання людини з метою підтримки оптимальних умов вирощування. Сюди відносяться штучне зрошення (полив) та внесення мінеральних чи органічних добрив під корінь або позакоренево. Ці заходи покликані штучно компенсувати дефіцит вологи та нутрієнтів (NPK), що виникає в процесі життєдіяльності культури.

Суттєвою особливістю предметної області є необхідність безперервної інтеграції біологічних процесів з фінансово-економічними показниками підприємства. Будь-яке антропогенне втручання (кожна окрема агрооперація) потребує використання матеріальних ресурсів (води, конкретних видів добрив), які мають свою ринкову вартість за одиницю об'єму чи маси. Крім того, проведення операції включає супутні операційні витрати (амортизація техніки, логістика, оплата праці).

У традиційній схемі господарювання агрономічний облік (журнали вимірювань, карти полів) відірваний від бухгалтерського або управлінського обліку фінансів. Фінансові підсумки підраховуються постфактум, після збору врожаю, що унеможливлює оперативне коригування витрат у разі несприятливих погодних умов. Проектована предметна область вимагає синхронного ведення обох контурів: у момент реєстрації або симуляції тижневого кроку розвитку

рослини, система повинна миттєво розраховувати накопичені фінансові витрати на дану секцію поля.

Для узагальненої оцінки ефективності управління посівами в аналізованій області використовуються спеціалізовані інтегральні індекси:

- Індекс вологозабезпеченості (WCI), що показує відхилення поточної вологи від норми, критичної для конкретної культури;
- Індекс азотного балансу (NBI), який сигналізує про рівень азотного голодування;
- Агро-економічний індекс (AEI), який відображає рентабельність проведених заходів через зіставлення динаміки приросту висоти культури із витраченими на цей приріст фінансовими коштами.

Таким чином, автоматизація даної предметної області вимагає створення цілісної інформаційної системи, здатної не лише накопичувати історичні дані щотижневих спостережень, а й гнучко моделювати взаємозв'язки між погодою, розвитком біомаси, зміною хімічного складу ґрунту та фінансовими витратами на підтримку життєдіяльності посівів сільськогосподарських культур.

1.2 Аналіз вимог до програмної системи

Процес розробки ефективного програмного забезпечення вимагає чіткого розуміння потреб кінцевих користувачів та умов експлуатації системи. Аналіз вимог є критичним етапом, що визначає функціональні межі та архітектурні особливості майбутнього продукту. Для системи відслідковування динаміки розвитку посівів «FieldSimulator» вимоги формуються на основі потреб двох основних груп стейкхолдерів (зацікавлених сторін): агрономів (безпосередніх користувачів) та керівництва/власників агропідприємства.

З точки зору керівництва підприємства, ключовою вимогою є забезпечення наскрізного економічного контролю. Система повинна автоматично трансформувати будь-які агротехнічні заходи у фінансові витрати, дозволяючи в реальному часі оцінювати рентабельність кожної окремої ділянки поля. Крім

того, критично важливою є вимога до надійності збереження історичних даних, що є основою для стратегічного планування на наступні сезони.

З точки зору агронома, система повинна виступати зручним інструментом для автоматизації рутини. Головними вимогами є мінімізація ручного введення даних, інтуїтивно зрозумілий інтерфейс для швидкої фіксації польових вимірювань та наявність математичної моделі, яка здатна симулювати розвиток посівів і завчасно попереджати про критичні стани ґрунту (дефіцит вологи або азоту).

На основі виявлених потреб стейкхолдерів було сформовано деталізований перелік функціональних та нефункціональних вимог.

Функціональні вимоги (Functional Requirements) Функціональні вимоги описують безпосередню поведінку системи, процеси обробки даних та доступні користувачеві сценарії використання. Програмна система повинна забезпечувати виконання наступних функцій:

1. Управління ієрархічною структурою господарства. Система повинна дозволяти створювати нові поля, задавати їх загальну площу та логічно розділяти їх на секції. Для кожної секції має бути передбачена можливість призначення сільськогосподарської культури з довідника та фіксації дати посіву.
2. Моніторинг та введення спостережень. Користувач повинен мати можливість вносити результати щотижневих польових вимірювань для обраної секції. До обов'язкових параметрів належать: поточна висота рослини, температура довкілля, вологість ґрунту, кількість опадів, а також рівень забезпеченості макроелементами (азот, фосфор, калій).
3. Математична симуляція розвитку (прогнозування). Система повинна містити алгоритм для симуляції стану культури на один тиждень вперед. Симуляція має базуватися на введених (або автоматично згенерованих кліматичних) показниках температури та опадів, розраховуючи зміну вологості, споживання нутрієнтів та приріст біомаси залежно від наявності сприятливих умов (знаходження у вікні активної вегетації).

4. Планування та облік агрооперацій. Перед запуском симуляції користувач повинен мати змогу додати список запланованих агротехнічних заходів (штучний полив, внесення конкретних видів добрив) із зазначенням норм внесення.
5. Економічна калькуляція. Система повинна автоматично розраховувати загальну вартість запланованих агрооперацій на основі довідкових цін на ресурси та додавати ці витрати до загального фінансового балансу секції.
6. Розрахунок інтегральних індексів. Після кожної реєстрації вимірювань або завершення симуляції користувач може перераховувати ключові індекси стану: індекс вологозабезпеченості (WCI), індекс азотного балансу (NBI) та агро-економічний індекс (AEI), візуалізуючи їх на відповідній вкладці.
7. Аутентифікація користувачів. Забезпечення базового рівня доступу до системи за допомогою механізму логіна та пароля для розмежування доступу до комерційних даних господарства.

Нефункціональні вимоги (Non-functional Requirements) Нефункціональні вимоги визначають атрибути якості системи, її обмеження та вимоги до середовища експлуатації.

1. Вимоги до інтерфейсу користувача (Usability). Графічний інтерфейс має бути побудований за принципами мінімалізму (Flat UI) з використанням сучасних компонентів. Всі поля для введення числових значень повинні мати систему захисту від некоректного вводу, що зменшить кількість помилок через людський фактор. Діалогові вікна повинні надавати чіткий зворотний зв'язок (повідомлення про успішне збереження або помилку).
2. Вимоги до надійності та цілісності (Reliability). З огляду на те, що система працює з фінансовими даними, часткове збереження інформації є неприпустимим. Запис результатів симуляції (оновлення показників ґрунту та списання коштів з балансу секції) повинен виконуватися виключно через механізм транзакцій реляційної бази даних . У разі апаратного збою під час збереження, система повинна автоматично відкотити зміни до початкового стану.

3. Вимоги до продуктивності (Performance). Обробка математичної моделі симуляції та розрахунок індексів повинні виконуватися в режимі реального часу (без помітних для користувача затримок), оскільки вони базуються на оптимізованих SQL-запитах до бази даних.
4. Вимоги до середовища (Environment). Система проектується як "товстий клієнт" (Desktop Application). Для її функціонування робоча станція користувача повинна працювати під управлінням операційної системи Windows (версії 10 або новішої) із встановленим середовищем виконання .NET Framework (версії не нижче 4.7.2). Для зберігання даних необхідна наявність локально або в мережі встановленої СКБД Microsoft SQL Server.

Реалізація наведених вимог дозволить створити надійний, зручний та економічно обґрунтований програмний продукт, що повністю задовольняє потреби стейкхолдерів у сфері моніторингу сільськогосподарських культур.

1.3 Моделювання предметної області

Для формалізації результатів системного аналізу та детального опису процесів, що відбуваються в аналізованій предметній області, необхідно розробити комплекс функціональних та поведінкових моделей. Об'єктно-орієнтоване моделювання дозволяє розглянути проектовану систему з різних точок зору, не перевантажуючи опис технічними деталями реалізації, які будуть розгорнуті на етапі безпосереднього проєктування.

На етапі аналізу предметної області першочерговим завданням є визначення меж системи та її взаємодії із зовнішнім середовищем. Ця задача вирішується за допомогою побудови діаграми прецедентів (Use Case Diagram) мови UML. Модель прецедентів фіксує функціональні можливості системи через взаємодію дійових осіб (акторів) та прецедентів (сценаріїв використання системи).

В розроблюваній інформаційній системі «FieldSimulator» виділено дві основні дійові особи (актори) (Рис. 1.1):

Рис. 1.1 Діаграма прецедентів інформаційної системи

1. **Агроном (Користувач)** - ключовий оператор системи, який безпосередньо взаємодіє з інтерфейсом для моніторингу полів, введення поточних метрик, планування агрозаходів та запуску імітаційного моделювання.
2. **Адміністратор системи / Керівник** - актор, який володіє повним набором прав користувача, а також додатково забезпечує налаштування системних параметрів, керування обліковими записами та редагування нормативно-довідкової інформації (ціни на ресурси, базові константи споживання для культур).

Зв'язок між акторами та бізнес-функціями системи відображає концептуальну схему автоматизації. До базових прецедентів системи належать: «Налаштування структури поля», «Фіксація стану посівів», «Проведення та облік агрозаходу», «Запуск щотижневої симуляції» та «Перегляд звітності та індексів».

Для глибшого розуміння логіки предметної області доцільно навести детальний текстовий аналіз (сценарії) основних прецедентів, що визначають динаміку системи:

- **Прецедент «Налаштування структури поля».**

  * Мета: Створення цифрового двійника територіальної структури господарства.

    - Передумови: Користувач успішно пройшов аутентифікацію в системі.
    - Основний сценарій: Користувач ініціює створення об'єкта «Поле», задає його площу. Далі виконується декомпозиція поля на «Секції». Для кожної секції з нормативно-довідкової бази обирається конкретна «Культура» та фіксується дата її висадки. Система перевіряє коректність даних та створює початковий логічний запис у базі даних, присвоюючи секції стартовий нульовий фінансовий баланс.

- **Прецедент «Фіксація стану посівів та запуск симуляції».**
    - Мета: Введення параметрів навколишнього середовища, реєстрація поточного стану рослин та прогнозування динаміки на крок вперед.
    - Передумови: Обрано конкретну секцію поля, для якої існує хоча б одне попереднє вимірювання (або стартові умови посіву).
    - Основний сценарій: Користувач вносить фактичні або прогнозні метеорологічні дані (середня температура за тиждень, об'єм опадів у міліметрах). За потреби, користувач додає заплановані на цей тиждень агрооперації (наприклад, полив в об'ємі V кубічних метрів або внесення азотних добрив масою M кілограмів). Після

натискання кнопки активації симуляційного кроку, система запускає обчислювальний модуль.

Поведінкова логіка обчислювального модуля (імітаційне моделювання процесу):

В основі життєвого циклу тижневого кроку лежить послідовний алгоритм обробки взаємопов'язаних факторів середовища. Спочатку система зчитує з бази даних останні відомі показники секції: поточну вологість ґрунту, залишкову концентрацію макроелементів N, P, K та висоту культури.

На першому етапі розраховується новий рівень вологості. Він є результатом суперпозиції базової вологості, природних опадів та штучного поливу (якщо агрооперація була запланована) за вирахуванням коефіцієнта природного випаровування (евапотранспірації), який нелінійно залежить від температурного фактору.

На другому етапі обчислюється динаміка поглинання поживних речовин. Рослина здатна абсорбувати азот, фосфор та калій з ґрунту лише за умови, що поточна температура знаходиться всередині біологічного вікна активної вегетації культури ($T_{min} \leq T \leq T_{max}$), а рівень вологості ґрунту не є критично низьким. Якщо умови задовольняються, показники N, P, K у ґрунті зменшуються пропорційно фазі росту культури, а якщо користувач вносив добрива - їхня маса додається до балансу хімічних елементів ґрунту.

На третьому етапі визначається лінійний приріст біомаси (висоти). Величина приросту є функцією від інтегрального коефіцієнта умов середовища: за наявності дефіциту хоча б одного з елементів (води, азоту, фосфору чи калію) ріст сповільнюється або повністю припиняється відповідно до закону мінімуму Лібіха.

На фінальному етапі симуляційного кроку виконується економічний розрахунок. Якщо в межах тижня проводилися агрооперації, система звертається до довідника ресурсів, вираховує їх сумарну вартість (Вартість = Об'єм*Ціна за одиницю) та здійснює операцію накопичення фінансових витрат секції (TotalExpenses). На основі отриманого приросту висоти та нових фінансових

витрат обчислюється інтегральний агро-економічний індекс (AEI), а також індекси WCI та NBI (Рис. 1.2).

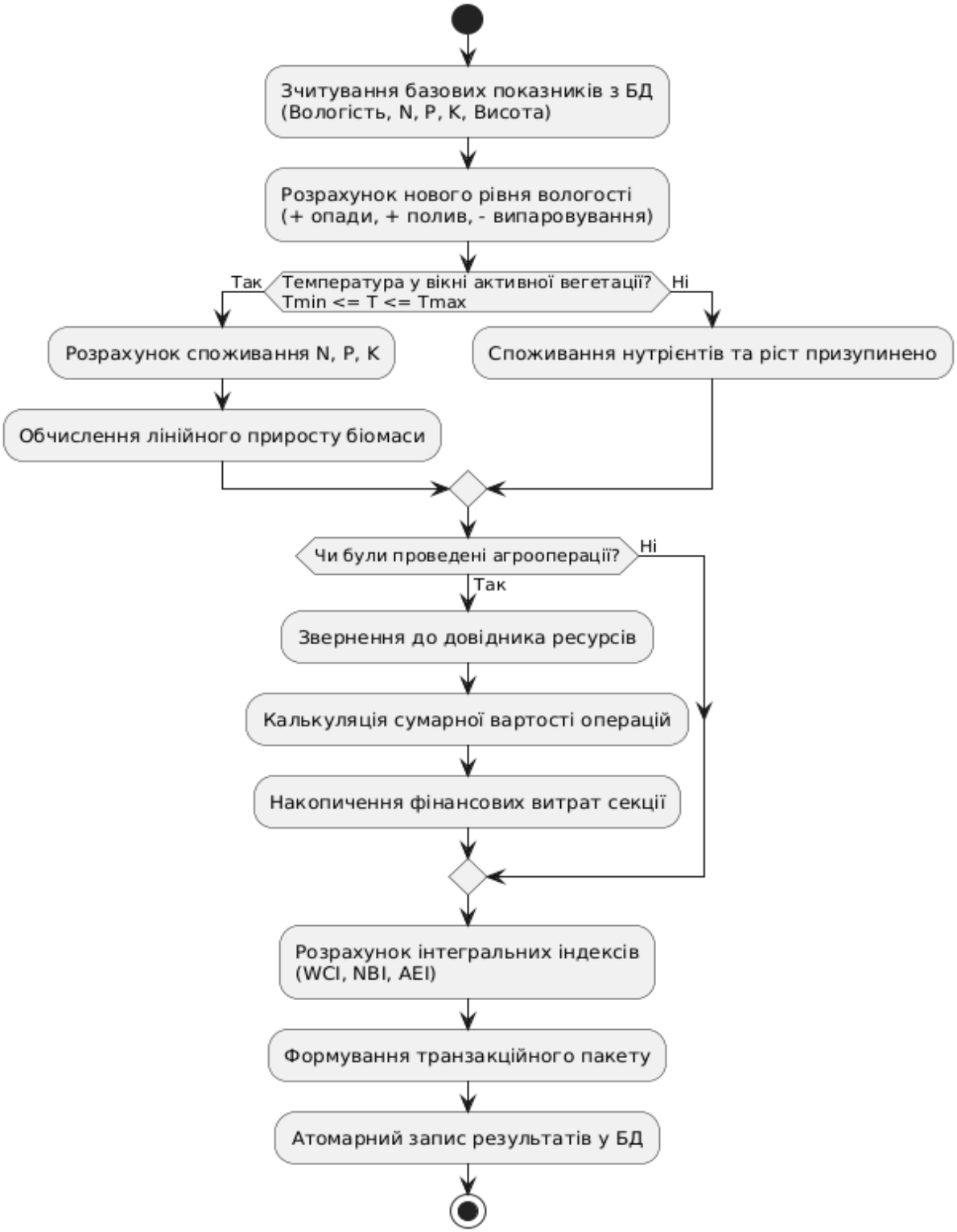


Рис. 1.2 Діаграма активності алгоритму симуляційного кроку

Для забезпечення надійності та виключення аномалій у системі, весь описаний ланцюжок розрахунків та фіксації результату в базі даних є атомарним

процесом. Моделювання бізнес-процесу завершується формуванням транзакційного пакету даних, який гарантує, що нові біометричні показники культури не будуть записані у разі виникнення помилок при калькуляції економічного контуру.

Таким чином, розроблений комплекс функціональних та поведінкових моделей чітко описує логічну структуру процесів моніторингу та симуляції, що дозволяє перейти від аналізу предметної області безпосередньо до детального проєктування архітектури та бази даних інформаційної системи.

1.4 Огляд інформаційних джерел та існуючих рішень

Розвиток інформаційних технологій в агропромисловому секторі призвів до формування окремого технологічного напрямку - точного землеробства (Precision Agriculture або AgTech). Аналіз сучасних літературних джерел та наукових публікацій свідчить, що більшість сучасних досліджень у цій галузі фокусується на використанні геоінформаційних систем (ГІС), даних супутникового моніторингу (для розрахунку вегетаційних індексів, таких як NDVI) та технологій Інтернету речей (IoT) для збору даних з полів у режимі реального часу.

Однак, незважаючи на потужний розвиток апаратного забезпечення, залишається актуальною проблема програмної обробки цих даних, зокрема їх інтеграції з фінансово-економічними показниками підприємства та побудови прогнозних (симуляційних) моделей розвитку культур.

Для визначення рівня вирішеності поставленої проблеми та виявлення вільних ніш було проведено аудит існуючих на ринку комерційних програмних рішень.

1. Програмний комплекс Cropio (належить компанії Syngenta) Система управління агровиробництвом, яка дозволяє здійснювати дистанційний контроль сільськогосподарських угідь, оперативно слідкувати за станом посівів та прогнозувати врожайність.

- **Переваги:** Високий рівень автоматизації завдяки інтеграції з супутниковими знімками високої роздільної здатності; наявність потужних інструментів для глобального аналізу вегетації на великих площах; автоматичне підтягування метеорологічних даних.
- **Недоліки:** Висока вартість підписки, що робить продукт економічно недоцільним для малих фермерських господарств. Система орієнтована переважно на глобальний моніторинг (макрорівень) і має обмежений функціонал для мікрорівневого (потижневого) моделювання стану окремої секції з урахуванням специфіки локального внесення конкретних добрив та їх миттєвого перерахунку у фінансові витрати. Крім того, система вимагає постійного високошвидкісного підключення до мережі Інтернет.

2. Хмарна платформа OneSoil Популярний безкоштовний (у базовій версії) додаток для точного землеробства, який дозволяє фермерам спостерігати за розвитком культур, створювати карти диференційованого посіву та внесення добрив.

- **Переваги:** Інтуїтивно зрозумілий та сучасний інтерфейс користувача; наявність безкоштовного базового функціоналу; зручні інструменти для польового скаутингу (додавання нотаток безпосередньо з поля через мобільний телефон).
- **Недоліки:** Оскільки система є повністю хмарною (SaaS), користувач не має повного контролю над конфіденційними даними свого господарства. Головний недолік у контексті розв'язуваної задачі - відсутність вбудованого математичного симулятора біологічних процесів. Система фіксує поточний стан за супутниковим знімком, але не дає змоги фермеру "програти" сценарій уперед (наприклад: "що буде з висотою культури та балансом азоту через тиждень, якщо температура впаде до 10 градусів, а я не внесу добрива?"). Економічний модуль обліку витрат також є мінімалістичним і не пов'язаним з біологічною моделлю.

3. Платформа Climate FieldView (від компанії Bayer) Комплексна платформа, яка збирає дані безпосередньо з техніки (тракторів, сівалок,

комбайнів) під час виконання польових робіт, формуючи точні карти врожайності та сівби.

- **Переваги:** Максимальна точність даних, оскільки вони збираються апаратними датчиками на місці; потужна аналітика результатів збору врожаю.
- **Недоліки:** Високий поріг входу - для повноцінної роботи системи потрібна закупівля та встановлення дорогого пропрієтарного обладнання (датчиків та моніторів) на сільськогосподарську техніку. Система є складною в налаштуванні і не передбачає легкого режиму "ручного" моделювання ситуацій для агронома, який працює без підключеної смарт-техніки.

Висновки щодо аналізу існуючих рішень. Проведений аудит інформаційних джерел та програмних продуктів показав, що ринок перенасичений глобальними хмарними ГІС-системами, які орієнтовані на збір масивів даних (Big Data) з супутників та техніки.

Разом з тим, існує дефіцит легких, десктопних локальних програмних рішень, які об'єднували б у собі простоту ведення щотижневого журналу спостережень, математичну модель симуляції життєдіяльності культури (з урахуванням вологи, температури та NPK) та покроковий економічний калькулятор. Більшість існуючих рішень не дозволяють агроному інтерактивно експериментувати з параметрами середовища та миттєво бачити вплив своїх рішень на агро-економічний індекс (AEI) рентабельності.

Саме тому розробка спроєктованої інформаційної системи «FieldSimulator», яка базується на математичному моделюванні та жорсткому транзакційному контролі фінансових витрат, є актуальною, науково обґрунтованою та має практичну цінність для цільової аудиторії.

1.5 Постановка завдання

На основі проведеного системного аналізу предметної області, вивчення потреб цільової аудиторії (стейкхолдерів, фермерів, агрономів) та виявлених

недоліків існуючих на ринку комерційних програмних продуктів (таких як надмірна складність інтерфейсу, висока вартість ліцензій, відсутність інтегрованого локального калькулятора витрат), виникає об’єктивна необхідність у створенні спеціалізованого програмного забезпечення.

Головною метою розробки є створення локальної інформаційної системи (десктопного додатку) **«FieldSimulator»** для автоматизації процесів моніторингу посівів, імітаційного моделювання їх розвитку під впливом динамічних кліматичних факторів та синхронного обліку фінансових витрат на впроваджувані агротехнічні заходи.

Для досягнення поставленої мети необхідно вирішити наступний комплекс інженерних, алгоритмічних та архітектурних завдань:

1. **Проєктування реляційної бази даних:** розробити оптимальну структуру даних з урахуванням вимог нормалізації (до третьої нормальної форми включно) у середовищі СКБД MS SQL Server. База даних повинна забезпечувати збереження ієрархічної структури сільськогосподарського підприємства (Поле $\rightarrow$ Секція $\rightarrow$ Культура), нормативно-довідкової інформації (дефолтні та кастомні ціни на ресурси: добрива, паливо, вода, засоби захисту рослин), а також деталізованої історії щотижневих вимірювань і виконаних технологічних операцій.
2. **Розробка алгоритму симуляції біологічних процесів:** програмно реалізувати комплексну математичну модель вегетації. Алгоритм на основі вхідних щотижневих параметрів (температура повітря, кількість опадів, поточний хімічний склад ґрунту) має розраховувати динаміку вологості, споживання ключових макроелементів (Азот - N, Фосфор - P, Калій - K) за законом мінімуму Лібіха, а також прогнозувати лінійний приріст біомаси та висоти культури з прив’язкою до фаз її розвитку.
3. **Реалізація економічного та аналітичного модуля:** створити підсистему калькуляції операційних витрат, яка автоматично перераховує обсяги використаних ресурсів (кількість на 1 га $\times$ площа секції) у грошовий еквівалент та акумулює фінансові витрати по кожній секції. Модуль також

має розраховувати інтегральні показники стану поля: Індекс вологозабезпеченості (WCI), Індекс азотного балансу (NBI) та Агро-економічний індекс (AEI).

4. **Забезпечення відмовостійкості та цілісності даних:** реалізувати надійний механізм захисту інформації від системних збоїв на рівні взаємодії додатка з СКБД за допомогою архітектури ADO.NET. Будь-який симуляційний крок або внесення агрооперації, що зачіпає декілька таблиць одночасно (лог операцій, баланс витрат секції, зміна параметрів ґрунту), мають виконуватися суворо всередині ізольованих SQL-транзакцій (відповідно до принципів ACID).
5. **Розробка ергономічного графічного інтерфейсу користувача:** спроєктувати UI/UX дизайн системи на базі технології Windows Forms із застосуванням сучасних UI-компонентів бібліотеки MetroFramework (Flat UI). Інтерфейс повинен мінімізувати кількість кліків для виконання рутинних операцій агронома та надавати наочну візуалізацію поточного стану секторів поля.

Вхідні та вихідні дані інформаційної системи

Для забезпечення коректного функціонування розроблюваного додатку необхідно чітко розмежувати інформаційні потоки, що циркулюють у системі.

Вхідними даними системи є:

- Геометричні та структурні параметри: загальна площа сільськогосподарських угідь, кількість ізольованих секцій (довідково), архітектура розподілу посівних площ.
- Агрономічні параметри: біологічні характеристики обраної культури, початковий стан ґрунту на момент старту симуляційного періоду (рівень вологості у відсотках, концентрація азоту, фосфору та калію в мг/кг).
- Метеорологічні фактори: щотижневі показники середньої температури навколишнього середовища, сума ефективних температур, обсяг природних опадів (мм).

- Технологічні параметри: обсяги внесених добрив (азотних, фосфорних, калійних або комплексних типу NPK 16:16:16), об'єми води для поливу, витрати дизельного палива та засобів захисту рослин (ЗЗР) у розрахунку на один гектар площі.

Вихідними даними (результатами роботи) системи є:

- Динамічні графіки та таблиці стану посівів: щотижневі звіти про лінійний ріст рослин та поточний дефіцит або профіцит поживних речовин у землі.
- Економічні метрики: сумарні фінансові витрати на обслуговування кожної окремої секції поля та загальні витрати по всьому господарству.
- Аналітичні висновки: розраховані коефіцієнти ефективності (WCI, NBI, AEI) для підтримки прийняття управлінських рішень щодо доцільності подальшого підживлення чи зрошення.

Технічні обмеження розробки

Програмний продукт розробляється як десктопний додаток, орієнтований на роботу в умовах обмеженого або відсутнього постійного інтернет-з'єднання (що є критичним для віддалених польових умов та польових офісів).

Система базується на платформі **.NET Framework версії 4.7.2** із використанням мови програмування **C# (версія 7.3)**. Взаємодія з базою даних здійснюється локально або через внутрішню мережу підприємства, що висуває підвищені вимоги до швидкодії SQL-запитів та оптимізації індексів у таблицях СКБД. Програма має функціонувати в операційних системах сімейства Windows (Windows 10, Windows 11) без необхідності встановлення додаткового високовартісного комерційного софту, окрім безкоштовної версії MS SQL Server Express.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Проєктування інформаційного забезпечення є фундаментальним етапом створення надійної програмної системи, оскільки від структури збереження даних залежить швидкість їх обробки, цілісність та гнучкість налаштувань. Для інформаційної системи «FieldSimulator» обрано реляційну модель даних, реалізація якої здійснюється за допомогою СКБД Microsoft SQL Server.

Основою для фізичної реалізації бази даних є розроблена логічна модель даних у вигляді ER-діаграми (Рис. 2.1). Вона відображає розширений набір сутностей предметної області, їхні атрибути та типи зв'язків між ними, враховуючи багатокористувацьку архітектуру та персоналізацію налаштувань.

Логічну модель можна розділити на кілька функціональних блоків сутностей:

1. **Блок користувачів та налаштувань:**
    - users - базова сутність для авторизації (зберігає логін та пароль).
    - UserSettings та ColorSettings - зберігають індивідуальні параметри інтерфейсу користувача (наприклад, стан розумних підказок enable_smart_prompts та кольорове маркування індексів).
2. **Блок ієрархії земельних угідь:**
    - fields - містить загальну інформацію про поля користувача (площа, прив'язка до клімату climate_id, локальні витрати на посів та збір).
    - section - підпорядкована сутність, що деталізує поле. Зберігає зв'язок з висадженою культурою, а також поточні розраховані індекси стану (AEI, NBI, GPI, CCI, WCI) та фінансові показники (витрати, очікуваний дохід, прибуток).

- field_backup - допоміжна сутність для резервного копіювання базової інформації про поля.

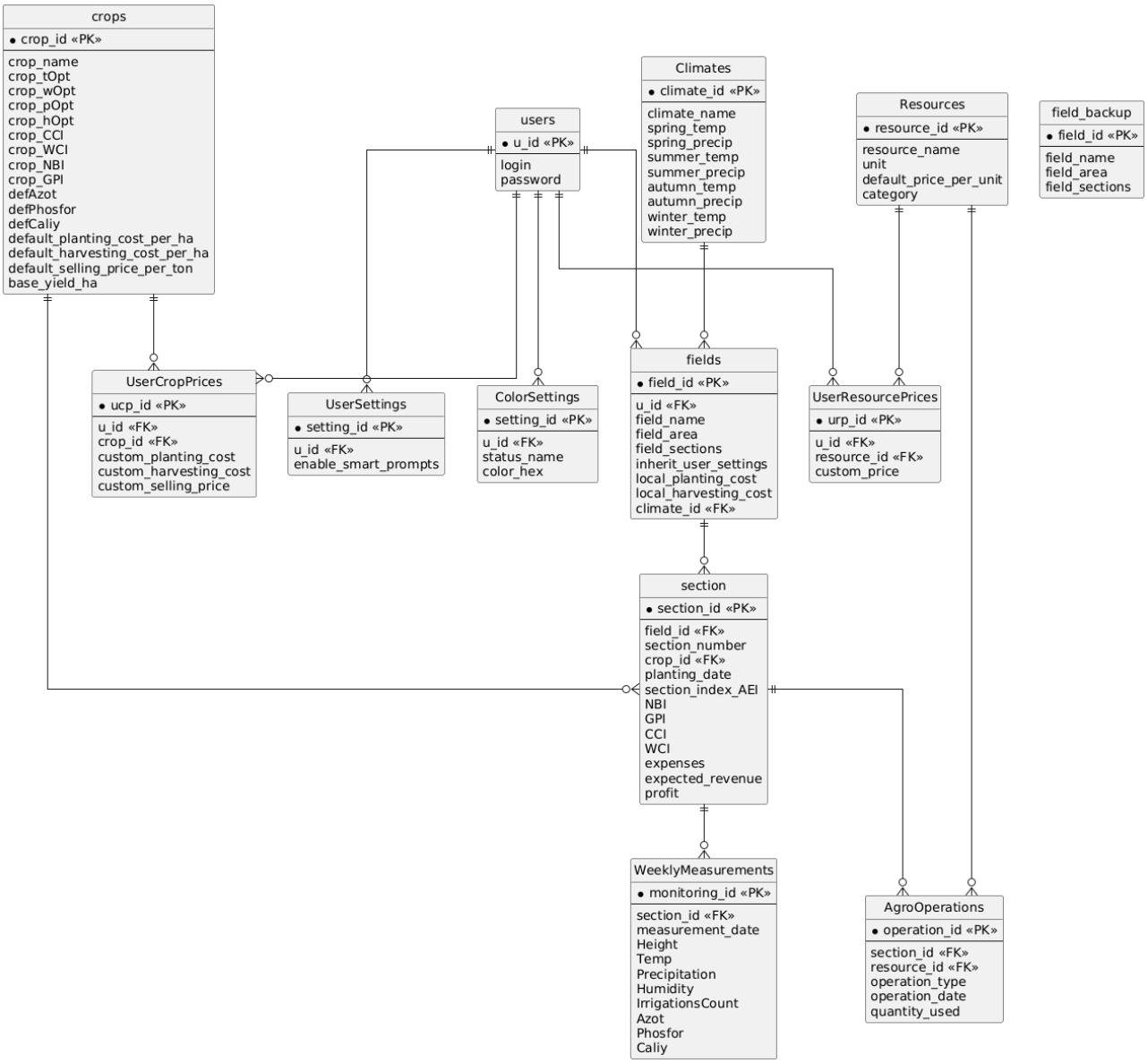


Рис. 2.1 Логічна модель бази даних у вигляді ER-діаграми

3. **Блок нормативно-довідкової інформації (Довідники):**
   - crops - розширений довідник сільськогосподарських культур із закладеними біологічними оптимумами (температури crop_tOpt, вологи, базовими потребами в NPK) та стандартними цінами.
   - Climates - довідник сезонних кліматичних норм (температури та опади для кожної пори року).

- Resources - довідник агроресурсів (добрива, вода) з базовою ціною за одиницю.

4. **Блок кастомізації економіки:**
   - UserCropPrices та UserResourcePrices - таблиці, що дозволяють конкретному користувачеві перевизначати стандартні системні ціни на культури та ресурси під реалії свого господарства.
5. **Транзакційний блок (Динаміка):**
   - WeeklyMeasurements - фіксує історичну динаміку розвитку на секції (висота, вологість, температура, рівень N, P, K на конкретну дату).
   - AgroOperations - логіює проведені агротехнічні заходи із зазначенням використаного ресурсу та його об'єму.

Доведення відповідності моделі вимогам реляційності та третій нормальній формі (3НФ): Спроєктована логічна модель повністю відповідає вимогам теорії реляційних баз даних та пройшла процес нормалізації до Третьої нормальної форми (3НФ), що унеможливлює виникнення аномалій модифікації, вставки та видалення даних:

1. Відповідність Першій нормальній формі (1НФ): Усі атрибути кожної сутності є атомарними. Наприклад, таблиця WeeklyMeasurements не містить масивів показників, кожне вимірювання формує окремий атомарний рядок. Кожна таблиця має унікальний первинний ключ (Primary Key), реалізований через сурогатні ідентифікатори (наприклад, u_id, section_id, monitoring_id).
2. Відповідність Другій нормальній формі (2НФ): Модель знаходиться у 1НФ і не має часткових залежностей. Усі неключові атрибути повноцінно залежать від усього первинного ключа. Оскільки в системі використовуються виключно прості (одинарні) первинні ключі, вимога 2НФ виконується автоматично.
3. Відповідність Третій нормальній формі (3НФ): Модель знаходиться у 2НФ і не містить транзитивних залежностей (жоден неключовий атрибут не залежить від іншого неключового атрибута). Наприклад, у таблиці

AgroOperations зберігається лише ідентифікатор ресурсу resource_id, а не його ціна чи назва. Якщо базова ціна ресурсу зміниться, її достатньо оновити лише в довіднику Resources (або в кастомних цінах UserResourcePrices), що підтверджує відсутність дублювання даних, високу цілісність та строгу відповідність моделі вимогам 3НФ.

2.2 Діаграма класів та кооперацій

Початковий етап проєктування прикладного програмного забезпечення інформаційної системи «FieldSimulator» базується на об'єктно-орієнтованому підході та починається з побудови загальної діаграми класів (Рис. 2.2). Ця діаграма визначає статичну структуру системи, її внутрішні сутності, їхні атрибути, методи та логічні зв'язки між ними (асоціації, композиції, агрегації та успадкування).

Для детального проєктування та реалізації окремих функціональних вимог, загальну діаграму класів було розбито на три окремі кооперації, кожна з яких описує ізольований сценарій роботи системи:

1. **Кооперація «Налаштування структури поля»**

Ця кооперація відповідає за створення користувачем архітектури свого господарства. Головний клас User (Агроном) через відношення агрегації управляє списком полів (Field). Клас Field зв'язаний жорстким відношенням композиції з класом Section, оскільки секції не можуть існувати окремо від поля. Кожна секція містить асоціативний потік на клас Crop для визначення поточної культури (Рис. 2.3).

2. **Кооперація «Фіксація стану посівів»**

Описує процес потижневого моніторингу параметрів. Клас Section виступає контейнером для композиційної колекції об'єктів WeeklyMeasurement (Виміри). Під час виклику методу обчислення відхилень, об'єкт виміру через відношення залежності звертається до еталонних параметрів класу Crop для аналізу біологічного стану рослини (Рис. 2.4).

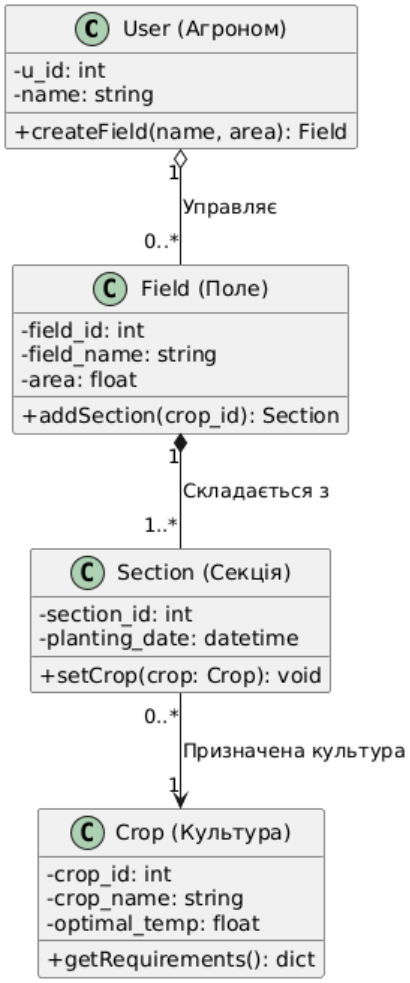


Рис. 2.3 Кооперація «Налаштування структури поля»

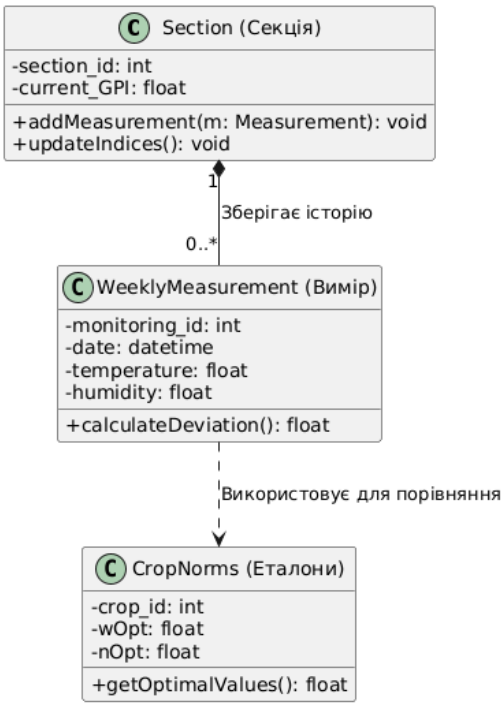


Рис. 2.4 Кооперація «Фіксація стану посівів»

3. Кооперація «Проведення та облік агрозаходу»

Відповідає за фіксацію дій на полі та динамічний перерахунок фінансових витрат. Клас Section фіксує у своєму журналі об'єкти абстрактного класу AgroOperation. Завдяки відношенню специфікації (успадкування), конкретні класи Irrigation (Полив) та Fertilization (Удобрення) реалізують власні алгоритми підрахунку вартості, використовуючи дані про вартість з об'єкта AgroResource (Рис. 2.5).

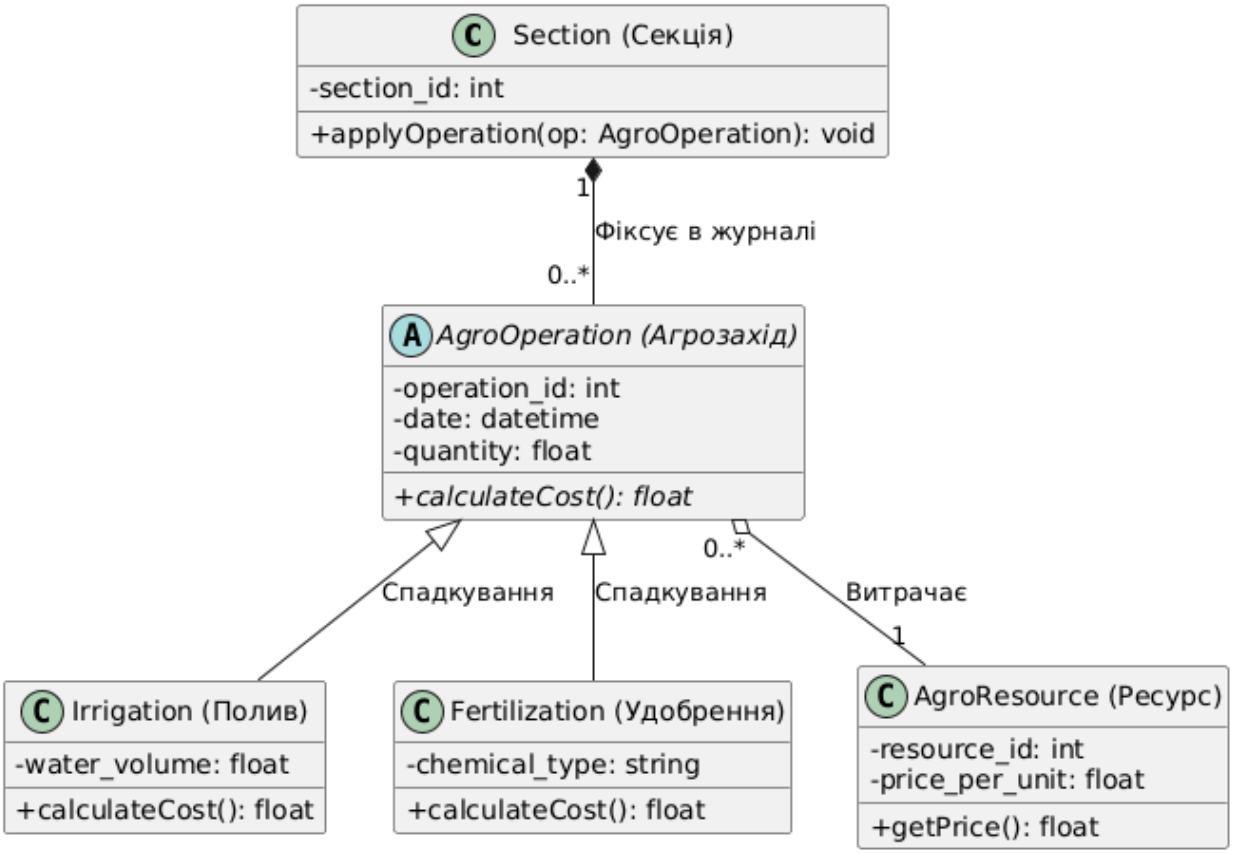


Рис. 2.5 Кооперація «Проведення та облік агрозаходу»

Обґрунтування шаблонів об’єктно-орієнтованого підходу

Відповідно до побудованих кооперацій, для реалізації прикладної програми було обрано такі шаблони (патерни) проєктування:

- **Шаблон Template Method :** Застосовано у кооперації «Проведення та облік агрозаходу». Абстрактний клас AgroOperation задає загальний скелет операції, а конкретні підкласи Irrigation та Fertilization реалізують

специфічний розрахунок вартості через поліморфне перевизначення методу calculateCost().

- **Шаблон Singleton :** Застосовується на рівні менеджерів даних для класу зв'язку з базою даних, що гарантує наявність лише одного стабільного підключення до СКБД під час виконання транзакцій у всіх трьох кооперацiях.

2.3 Діаграма пакетів

Побудова діаграми пакетів UML надає можливість представити високорівневу архітектуру програмної системи, логічно розбити її на окремі підсистеми та розмістити у них відповідні кооперації класів. Для забезпечення гнучкості розробки, простоти тестування та супроводу інформаційної системи «FieldSimulator», її архітектуру було спроєктовано за архітектурним принципом трирівневого розділення шарів (3-Tier Architecture).

Уся система розподілена на чотири основні внутрішні пакети (простори імен у середовищі розробки .NET/C#) та один зовнішній пакет залежностей:

1. **FieldSimulator.UI (Пакет користувацького інтерфейсу):** Цей пакет є верхнім рівнем системи і містить класи графічних екранних форм (MainForm, SimulationForm, AuthorizationForm та EditFieldForm). Його головна задача - збір вхідних даних від користувача (агронома/керівника) та візуалізація результатів обчислень. Він має пряму залежність від пакету сервісів для передачі керування та пакету моделей для відображення даних.
2. **FieldSimulator.Services (Пакет бізнес-логіки):** Центральний пакет архітектури, де зосереджено обчислювальні алгоритми та правила предметної області. Він містить класи-сервіси SimulationService (реалізує математичне моделювання росту посівів та розрахунок індексів AEI, NBI, WCI), SectionService (керування секціями та фінансами) та AuthService (перевірка прав доступу). Сервіси є повністю ізольованими від інтерфейсних компонентів.

3. **FieldSimulator.Models (Пакет моделей даних):** Містить легковагові класи сутностей (User, Field, Section, AgroOperation), які повторюють логічну структуру предметної області. Ці класи використовуються як уніфіковані контейнери (DTO) для транспортування інформації між усіма шарами системи.
4. **FieldSimulator.DataAccess (Пакет доступу до даних):** Нижній інфраструктурний рівень, відповідальний за взаємодію з базою даних MS SQL Server. Класи DatabaseConnection (що реалізує патерн Singleton) та SqlHelper інкапсулюють у собі створення підключень, запуск збережених процедур та управління SQL-транзакціями.

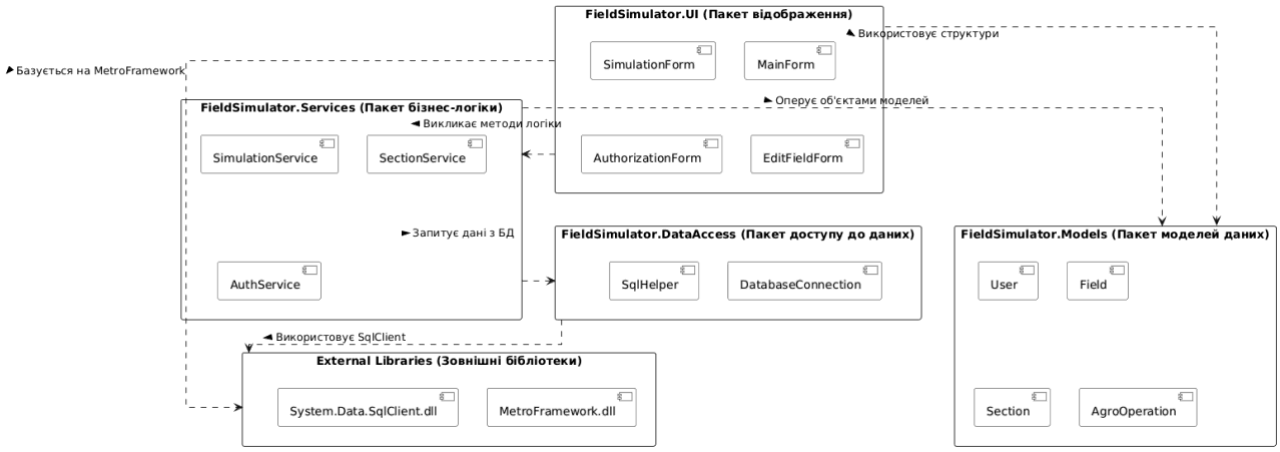


Рис. 2.6 Діаграма пакетів проектованої системи

Аналіз залежностей та архітектурних переваг: Як показано на Рис. 2.6, зв'язки між пакетами спрямовані строго зверху вниз, що мінімізує зачеплення (low coupling) компонентів додатка. Шари користувацького інтерфейсу (UI) не мають жодного прямого доступу до шару даних (DataAccess) чи виконання прямих SQL-запитів - будь-яка дія обов'язково проходить крізь валідацію та обробку в пакеті Services.

Також відображено використання зовнішнього пакету бібліотек (External Libraries), куди входять MetroFramework.dll (забезпечує графічний інтерфейс користувача) та System.Data.SqlClient.dll (забезпечує низькорівневий драйвер комунікації з сервером баз даних). Такий модульний поділ гарантує, що зміна дизайну однієї з форм у пакеті UI ніяк не вплине на математичний алгоритм

симуляції в пакеті Services, а перехід на іншу структуру збереження даних вимагатиме модифікації виключно пакету DataAccess.

2.4 Діаграма компонентів

Діаграмою компонентів завершується етап проєктування прикладної програми інформаційної системи «FieldSimulator». Цей тип діаграми UML призначений для візуалізації реальної фізичної структури програмного забезпечення: файлів, бібліотек динамічного компонування (.dll), конфігураційних артефактів та виконуваних модулів (.exe), а також архітектурних зв’язків між ними під час розгортання.

Розроблений програмний комплекс розгортається за схемою «товстого клієнта» (Thick Client) і на фізичному рівні складається з наступних вузлів та компонентів (Рис. 2.7):

1. **FieldSimulator.exe (Головний виконуваний модуль):** Центральний скомпільований файл системи, який запускається в середовищі ОС Windows. Він містить у собі весь байт-код користувацького інтерфейсу та бізнес-логіки.
2. **App.config (Конфігураційний артефакт):** Файл у форматі XML, що лежить у кореневій папці програми. Головний модуль звертається до нього при старті для зчитування глобальних налаштувань та конфігураційного рядка підключення до бази даних (Connection String).
3. **MetroFramework.dll та MetroFramework.Design.dll:** Зовнішні бібліотеки динамічного розширення, які імпортуються головним модулем для відмальовування сучасного графічного інтерфейсу (Flat UI).
4. **System.Data.SqlClient.dll (Провайдер доступу до даних):** Низькорівнева бібліотека платформи .NET, яка інкапсулює логіку взаємодії з СКБД. Вона бере на себе формування мережевих пакетів для відправки SQL-запитів.

5. **Вузол MS SQL Server Engine:** Серверний процес СКБД, який обробляє запити. Сама ж база даних FieldSimulator_DB фізично декомпозована на два обов'язкових артефакти:
    - FieldSimulator.mdf - головний файл даних (Primary Data File), де зберігаються всі таблиці (users, fields, section тощо).
    - FieldSimulator_log.ldf - файл журналу транзакцій (Transaction Log File), який гарантує цілісність даних при записі результатів симуляції.

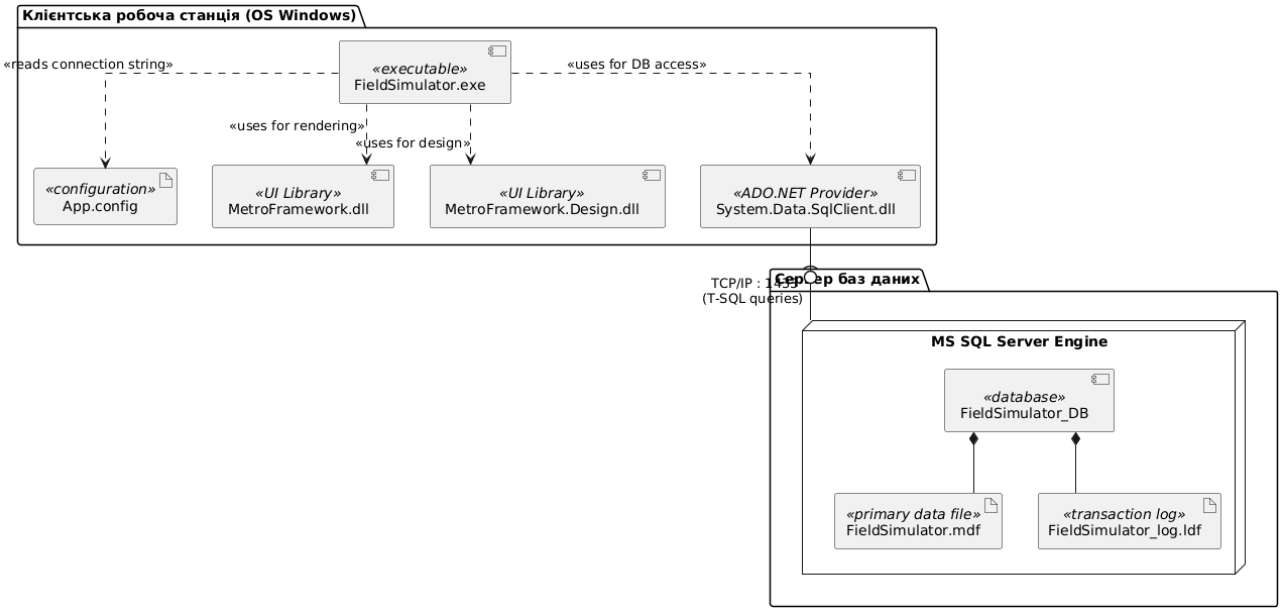


Рис. 2.7 Діаграма компонентів програмного забезпечення

Взаємодія між клієнтською частиною додатку (компонент SqlClient) та сервером баз даних реалізована за допомогою стандартного мережевого протоколу TCP/IP через порт 1433 (або механізму локальних Named Pipes). Комунікація відбувається шляхом передачі T-SQL інструкцій у рамках технології ADO.NET, що забезпечує надійне збереження даних під час фіксації агрозаходів та розрахунку індексів.

3. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

Для успішної реалізації логічної моделі даних інформаційної системи «FieldSimulator» першочерговим завданням є вибір та обґрунтування системи управління інформаційним забезпеченням. Враховуючи архітектуру системи, складність бізнес-процесів моделювання посівів та необхідність збереження великих обсягів транзакційних даних (потижневих вимірів), використання звичайної файлової системи є недоцільним через відсутність механізмів контролю цілісності, низьку швидкість пошуку та ризики руйнування даних при паралельному доступі.

Для управління інформаційною базою проєкту було обрано систему управління базами даних (СУБД) **Microsoft SQL Server**. За своєю внутрішньою структурою ця СУБД є суворо **реляційною**, що ідеально відповідає спроєктованій у підрозділі 2.1 логічній ER-моделі, яка базується на таблицях, первинних і зовнішніх ключах та декларативних обмеженнях цілісності.

Вибір саме СКБД MS SQL Server зумовлений такими архітектурними чинниками:

1. Локація та розгортання: СУБД може функціонувати як у централізованому клієнт-серверному режимі (на виділеному сервері підприємства чи в хмарі), так і локально (MS SQL Server Express) на робочій станції агронома, що забезпечує автономність роботи додатка «FieldSimulator» без обов'язкового підключення до глобальної мережі Інтернет.
2. Продуктивність та масштабованість: Наявність потужного процесора індексації та оптимізатора запитів дозволяє миттєво виконувати аналітичні вибірки за історичними даними потижневих моніторингів для тисяч секцій одночасно.

3. Повна інтеграція зі стеком .NET: СКБД має рідну підтримку з боку технологій Microsoft, що гарантує максимальну швидкість комунікації через драйвер SqlClient та стабільне управління SQL-транзакціями.
4. Підтримка мови T-SQL (Transact-SQL): Наявність розширення стандартної мови SQL дозволяє перенести частину обчислювальної логіки (наприклад, агрегацію фінансових витрат чи автоматичне резервне копіювання в таблицю field_backup) безпосередньо на рівень серверу у вигляді збережених процедур та тригерів.

Перехід від логічної структури (концептуальних сутностей) до фізичної структури бази даних здійснюється шляхом прямого відображення сутностей у реляційні таблиці з визначенням конкретних фізичних типів даних для кожного атрибута та генерацією DDL-коду (Data Definition Language) мовою SQL.

3.2 Розробка інформаційної бази

Перехід від теоретичного проєктування до практичного розгортання системи «FieldSimulator» вимагає трансформації логічної ER-моделі у фізичну структуру реляційної бази даних FieldSimulatorDB під керуванням СКБД MS SQL Server. На цьому етапі критично важливо забезпечити точність відображення сутностей, оптимізацію типів даних для економії дискового простору та побудову декларативних обмежень цілісності.

Для повного розгортання інформаційної бази на сервері було розроблено та виконано комплексний SQL-скрипт (DDL - Data Definition Language). Черговість створення таблиць суворо регламентована архітектурою зв'язків: спочатку створюються незалежні таблиці-довідники, а потім - підпорядковані сутності, що містять зовнішні ключі (Foreign Keys).

У ДОДАТКУ А наведено повний та автентичний SQL-код створення інформаційної бази системи.

Обґрунтування структури та специфікація типів даних

Під час переходу до фізичного рівня було проведено ретельний підбір системних типів даних СКБД для оптимізації обчислювальних процесів бізнес-логіки додатка:

- **Фінансові показники:** Для атрибутів expenses, expected_revenue, profit, а також усіх базових і кастомних цін (default_price_per_unit, custom_selling_price тощо) обрано тип DECIMAL(18,2). Використання типів із плаваючою комою (FLOAT/REAL) для грошових розрахунків є неприпустимим через накопичення помилок заокруглення в процесі арифметичних операцій агрегації. Тип DECIMAL(18,2) гарантує збереження точної фіксованої коми (2 знаки після коми).
- **Текстові поля:** Для назв полів, культур, кліматичних зон та агроресурсів використано тип NVARCHAR, що підтримує кодування Unicode. Це забезпечує коректне відображення кириличних символів в українськомовному інтерфейсі користувача.
- **Індекси та виміри стану:** Показники вегетаційних індексів (AEI, NBI, GPI, CCI, WCI) типізовані як DECIMAL(5,4) для забезпечення максимальної математичної точності симуляції. Температурні оптимуми, рівні вологи та об'єми елементів азоту/фосфору/калію (NPK) переведено у формат DECIMAL(8,2) або DECIMAL(5,2). Це дозволило уніфікувати підхід до чисел із плаваючою крапкою та уникнути непередбачуваної поведінки типу FLOAT при порівнянні значень у бізнес-логіці.

Забезпечення цілісності та оптимізація швидкодії

Для захисту інформаційної бази від логічного руйнування та забезпечення максимальної швидкодії було реалізовано такі механізми на рівні СКБД:

1. **Декларативні обмеження зв'язків (Referential Integrity):** На більшості зв'язків активовано правило ON DELETE CASCADE (Каскадне видалення). Наприклад, у разі видалення користувача з системи, сервер автоматично видалить його індивідуальні налаштування кольорів, кастомні ціни та всі його поля. У свою чергу, видалення поля автоматично каскадно

очистить інформацію про його секції, потижневі виміри та журнал операцій, запобігаючи появі «записів-сиріт».

2. **Індексація даних:** СКБД MS SQL Server автоматично створює кластеризовані індекси (Clustered Indexes) за первинними ключами таблиць. Проте, враховуючи специфіку транзакційного навантаження системи «FieldSimulator», де найчастішими операціями є вибірка вимірів для графіків та аудит операцій, на рівні фізичного розгортання генеруються додаткові некластеризовані індекси (Non-Clustered Indexes) за зовнішніми ключами: WeeklyMeasurements(section_id) та AgroOperations(section_id). Це забезпечує стабільну швидкість відгуку інтерфейсу при зростанні обсягу моніторингових даних.

Для фіксації повної фізичної архітектури розгорнутої бази даних нижче наведено діаграму її схему БД, яка була згенерована безпосередньо у середовищі СКБД MS SQL Server.

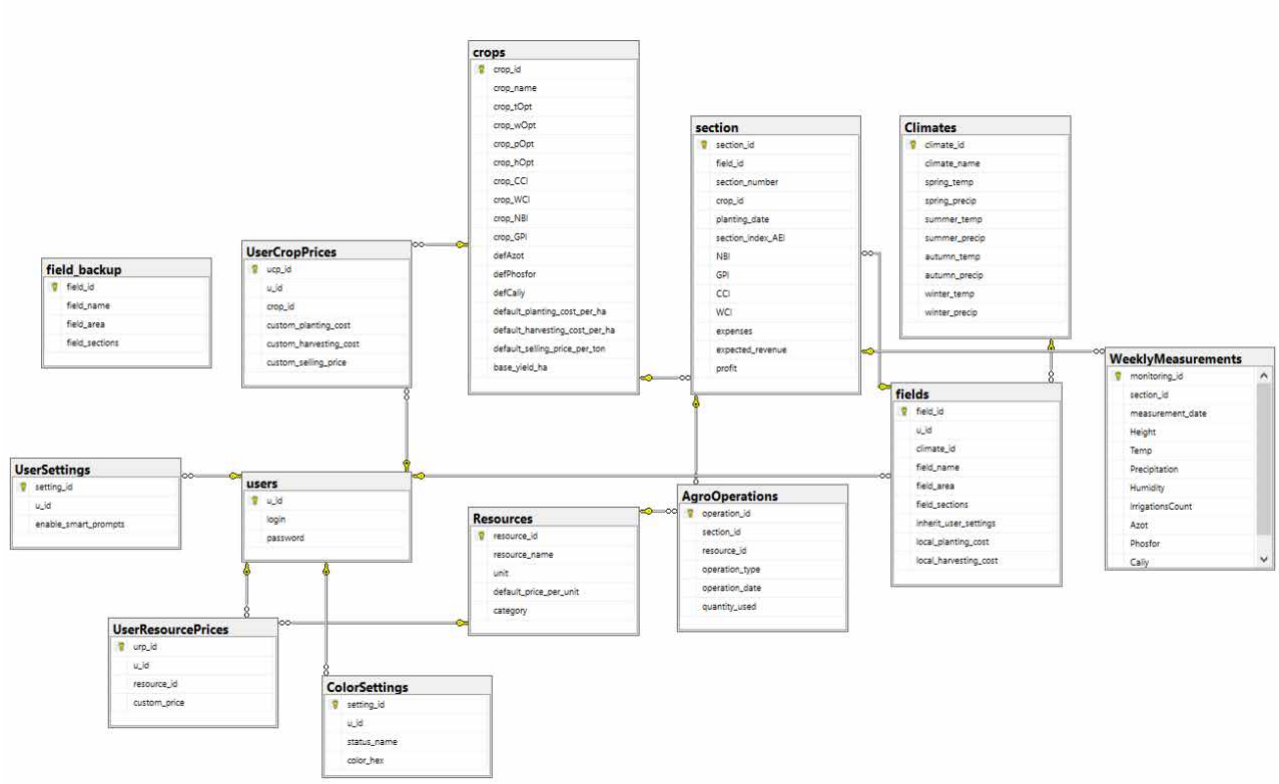


Рис. 3.1 Схема бази даних FieldSimulator_DB у середовищі MS SQL Server

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментальних засобів розробки прикладного програмного забезпечення для інформаційної системи «FieldSimulator» є результатом спільного обґрунтованого рішення студента-розробника та наукового керівника, виходячи з функціональних вимог до додатку, необхідності забезпечення високої швидкодії математичного моделювання та створення ергономічного інтерфейсу користувача.

Для реалізації різних частин і шарів прикладного програмного забезпечення було обрано та інтегровано такі технології та середовища розробки:

1. **Мова програмування C#:** Обрана як головна мова розробки бізнес-логіки та інтерфейсу. C# є суворо типізованою об'єктно-орієнтованою мовою, що забезпечує високу надійність коду, безпечне управління пам'яттю за допомогою вбудованого збирача сміття (Garbage Collector) та підтримку сучасних концепцій ООП (поліморфізм, інкапсуляція, успадкування), які активно використовуються в кооперaціях системи. Наявність технології LINQ (Language Integrated Query) дозволяє гнучко та швидко оперувати колекціями об'єктів моделей даних у пам'яті.
2. **Платформа .NET Framework / .NET:** Виступає в ролі базового середовища виконання (Runtime), надаючи величезну стандартну бібліотеку класів (Base Class Library) для роботи з колекціями, потоками даних, криптографічного шифрування паролів та мережевої комунікації.
3. **Технологія Windows Forms:** Використана для побудови клієнтського графічного додатка. Вибір зумовлений наявністю зручного візуального дизайнера (WYSIWYG), високою швидкістю рендерингу вікон на робочій станції користувача та низькими вимогами до апаратних ресурсів комп'ютера, що важливо для розгортання ПЗ на підприємствах аграрного сектору.

4. **Компонентна бібліотека MetroFramework (MetroModernUI):** Інтегрована через менеджер пакетів NuGet для розширення стандартних можливостей Windows Forms. Бібліотека дозволяє створювати інтерфейси у сучасному стилі Flat UI (чисті лінії, адаптивна палітра кольорів, плавні переходи), що кардинально покращує ергономіку, спрощує сприйняття графіків і таблиць індексів посівів користувачем та робить додаток конкурентоспроможним.
5. **Технологія ADO.NET (бібліотека System.Data.SqlClient):** Використовується як технологічний міст між C#-кодом та базою даних. Вона забезпечує швидке виконання параметризованих SQL-запитів, роботу з курсорами даних (SqlDataReader) та відправку пакетних транзакцій на сервер, захищаючи додаток від уразливостей типу SQL-ін'єкцій.
6. **Інтегроване середовище розробки Microsoft Visual Studio:** Використано як єдиний інструментальний простір, що об'єднує в собі редактор коду з інтелектуальним підказуванням (IntelliSense), засоби рефакторингу, вбудований графічний дизайнер форм та потужний дебагер (налагоджувальник) для покрокового контролю змінних під час симуляції росту рослин.

Така комбінація інструментів дозволила успішно розділити систему на незалежні пакети відповідно до архітектурної моделі проєкту, забезпечуючи високу швидкість розробки та стабільність функціонування готового програмного продукту.

3.4 Алгоритмізація та програмування програмних модулів

На етапі програмування програмних модулів інформаційної системи «FieldSimulator» було реалізовано бізнес-логіку обробки даних, механізми безпечного доступу до бази даних та алгоритми математичного моделювання стану посівів. Кодування здійснювалося мовою C# з використанням об'єктно-орієнтованих принципів (SOLID) та патернів проєктування.

Реалізація інфраструктурного рівня доступу до даних

В основі надійної роботи будь-якої клієнт-серверної програми лежить механізм управління підключеннями до бази даних. Для уникнення дублювання конфігураційних налаштувань та централізованого управління об'єктами SqlConnection, у системі було розроблено статичний клас Database, який виконує роль фабрики підключень.

Цей клас зберігає рядок підключення (Connection String) та інкапсулює метод ініціалізації з'єднання. Використання фабричного методу GetConnection(), який щоразу повертає новий екземпляр SqlConnection, є оптимальним патерном для технології ADO.NET, оскільки вона підтримує механізм пулінгу підключень (Connection Pooling). Це дозволяє економити ресурси сервера, та не утримувати з'єднання з БД постійно відкритим (Рис. 3.2).

```
using System.Data.SqlClient;

namespace FieldSimulator
{
    public static class Database
    {
        private static readonly string connectionString =
            @"Data Source=DIMA;Initial Catalog=FieldSimulatorDB;Integrated Security=True;";

        public static string ConnectionString => connectionString;

        public static SqlConnection GetConnection()
        {
            return new SqlConnection(connectionString);
        }
    }
}
```

Рис. 3.2 Реалізація класу управління підключенням до СКБД

Алгоритмізація процесу авторизації та управління сесіями

Модуль авторизації є точкою входу в інформаційну систему і відповідає за ідентифікацію агронома або адміністратора. Для візуалізації інтерфейсу використано клас Authorization, успадкований від MetroForm.

Головним завданням цього модуля є безпечна перевірка введених облікових даних. З метою захисту від атак типу SQL-ін'єкцій (SQL Injection) в алгоритмі суворо заборонено пряму конкатенацію рядків у SQL-запиті. Замість цього використовуються параметризовані запити колекції Parameters об'єкта SqlCommand.

Крім того, для оптимізації швидкодії замість завантаження всього рядка даних через SqlDataAdapter використовується метод ExecuteScalar(), який повертає лише одне значення (ідентифікатор користувача u_id), що суттєво зменшує навантаження на оперативну пам'ять.

Алгоритм роботи модуля авторизації наведено на блок-схемі нижче (Рис. 3.3.).

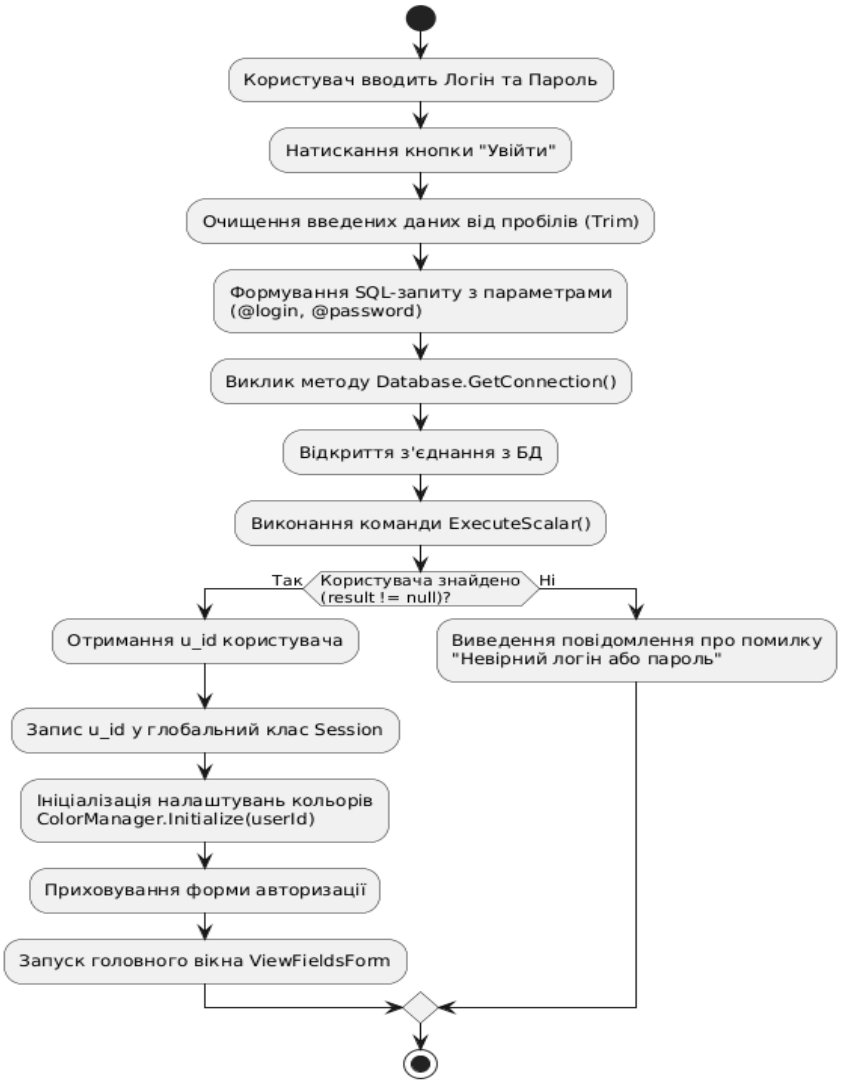


Рис. 3.3 Блок-схема алгоритму авторизації користувача

Фрагмент програмної реалізація перевірки облікових даних на ведений на Рис. 3.4.

```
private int GetUserId(string login, string password)
{
    // Використання блоку using гарантує автоматичне закриття з'єднання
    // та вивільнення некерованих ресурсів після завершення роботи
    using (SqlConnection connection = Database.GetConnection())
    {
        try
        {
            connection.Open();
            string query = "SELECT u_id FROM users WHERE login = @login AND password = @password";
            SqlCommand command = new SqlCommand(query, connection);

            // Захист від SQL-ін'єкцій через використання параметрів
            command.Parameters.AddWithValue("@login", login);
            command.Parameters.AddWithValue("@password", password);

            // ExecuteScalar використовується для швидкого отримання одного значення
            object result = command.ExecuteScalar();
            if (result != null && result != DBNull.Value)
            {
                return Convert.ToInt32(result);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка з'єднання: " + ex.Message);
        }
    }
    return -1; // Повернення маркера помилки, якщо користувача не знайдено
}
```

Рис. 3.4 Програмна реалізація перевірки облікових даних

У разі успішної перевірки система записує отриманий ідентифікатор у глобальний статичний клас Session, що дозволяє іншим формам системи ідентифікувати поточного суб'єкта, та ініціалізує його персональні налаштування інтерфейсу (клас ColorManager).

Алгоритмізація математичної моделі розвитку посівів

Ядром інформаційної системи «FieldSimulator» є обчислювальний модуль SimulationService, який відповідає за імітаційне моделювання динаміки росту культури та зміну показників ґрунту на щотижневій основі. Для забезпечення наукової достовірності результатів симуляції, алгоритм базується на загальноприйнятих агрономічних математичних моделях.

Процес симуляції (метод RunWeeklySimulation) враховує такі наукові підходи:

1. **Моделювання водного балансу:** Зміна рівня вологості ґрунту розраховується на основі спрощеної моделі евапотранспірації (FAO-56 Penman-Monteith). Новий рівень вологи є сумою попереднього стану, кількості опадів та штучного поливу, від якої віднімається обсяг випаровування (який лінійно залежить від підвищення температури повітря понад +5°C).
2. **Моделювання приросту біомаси (висоти):** Базується на концепції суми активних температур (Growing Degree Days - GDD). Алгоритм фіксує приріст лише в ті тижні, коли середня температура перевищує біологічний мінімум (базову температуру +10°C).
3. **Моделювання споживання макроелементів (NPK):** Згідно із Законом мінімуму Лібіха, обсяг споживання азоту, фосфору та калію розраховується динамічно і є прямо пропорційним до обсягу тижневого приросту біомаси. Якщо рослина перебуває у стані стресу (дефіцит вологи або азоту), ріст блокується, а споживання елементів різко знижується.

Загальний покроковий алгоритм розрахунку симуляційного кроку наведено на блок-схемі (Рис. 3.5.).

Транзакційна обробка та фінансовий облік агрозаходів

Важливою частиною функціоналу є фіксація діяльності агронома. За цю логіку відповідає клас AgroOperationService. Програмна реалізація методу ExecuteOperation вирішує одразу три задачі в рамках однієї атомарної операції:

1. **Логування:** записує факт проведення операції (полив, удобрення) в таблицю AgroOperations.

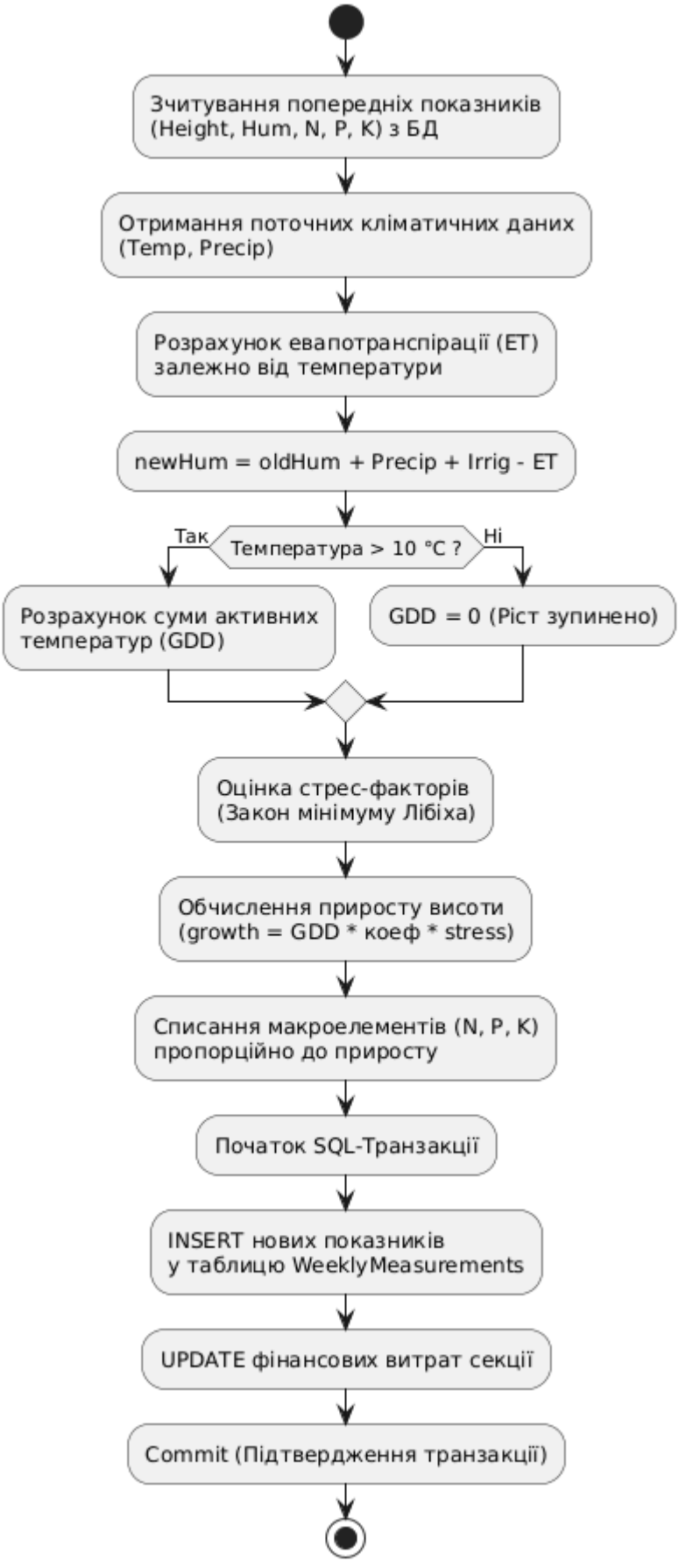


Рис. 3.5 Блок-схема алгоритму щотижневої симуляції стану поля

2. **Економічний аудит:** автоматично розраховує загальну вартість заходу за формулою:

$$Вартість = Норма\_внесення * Площа\_секції * Ціна\_ресурсу.$$

Ціна ресурсу динамічно визначається з урахуванням кастомних налаштувань користувача (таблиця UserResourcePrices). Отримана сума додається до загальних витрат (expenses) секції.

3. **Модифікація середовища:** залежно від типу операції (OperationType), система миттєво підвищує рівень вологи або відповідного макроелемента (N, P, K) в останньому записі моніторингу, що вплине на наступний крок симуляції. Фрагмент коду реалізації методу виконання агрозаходу наведений на Рис. 3.6.

```
public static void ExecuteOperation(int sectionId, int resourceId, OperationType opType, decimal quantityPerHa, decimal areaHa, decimal pricePerUnit)
{
    decimal totalCost = quantityPerHa * areaHa * pricePerUnit;

    using (SqlConnection conn = Database.GetConnection())
    {
        conn.Open();
        using (SqlTransaction trans = conn.BeginTransaction())
        {
            try
            {
                // 1. Запис в історію операцій
                string logQuery = "INSERT INTO AgroOperations (section_id, resource_id, operation_type, operation_date) VALUES (@SecId, @ResId, @OpType, GETDATE())";
                /* ... виконання команди логування ... */

                // 2. Списання фінансових коштів
                string expQuery = "UPDATE section SET expenses = ISNULL(expenses, 0) + @Cost WHERE section_id = @SecId";
                /* ... виконання команди списання витрат ... */

                // 3. Динамічний перерахунок показників ґрунту
                string columnToUpdate = GetColumnByOperation(opType); // Визначення стовпця (Azot, Phosfor, Caliy, Humidity)
                string updateDataQuery = $"UPDATE WeeklyMeasurements SET {columnToUpdate} = {columnToUpdate} + @Boost WHERE monitoring_id = (SELECT TOP 1 monitoring_id FROM WeeklyMeasurements WHERE section_id = @SecId ORDER BY measurement_date DESC)";
                /* ... виконання оновлення останнього моніторингу ... */

                trans.Commit();
            }
            catch
            {
                trans.Rollback();
                throw;
            }
        }
    }
}
```

Рис. 3.6 Фрагмент коду реалізації методу виконання агрозаходу

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Етап впровадження програмного продукту обов'язково починається з його тестування. Тестування програмного забезпечення - це процес перевірки відповідності розробленої системи заявленим функціональним та нефункціональним вимогам, а також виявлення дефектів (багів) перед передачею продукту кінцевому користувачеві. Метою тестування є забезпечення високої надійності, відмовостійкості та зручності використання інформаційної системи.
З огляду на архітектуру десктопного додатка «FieldSimulator» та наявність розвиненого графічного інтерфейсу, основним методом перевірки було обрано функціональне тестування за методом «чорного ящика» (Black-box testing). За цього підходу тестувальник не взаємодіє з вихідним кодом програми, а перевіряє її поведінку виключно через інтерфейс користувача, подаючи різні вхідні дані та аналізуючи отриманий результат.

Процес тестування формалізується за допомогою створення тестових сценаріїв (тест-кейсів), які покривають критичні бізнес-процеси системи. Для ілюстрації процесу проведення тестування розробленої системи нижче наведено Рис. 4.1 з основними тест-кейсами.

За результатами проведеного тестування (Рис. 4.2 – Рис. 4.6) можна констатувати, що розроблена система стабільно обробляє коректні дані та коректно реагує на виняткові ситуації, що підтверджує її готовність до дослідної експлуатації.

№	Назва тест-кейсу (Дія)	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Авторизація існуючого користувача	Логін: admin Пароль: 12345	Система перевіряє дані в БД, записує u_id в сесію та відкриває головне вікно ViewFieldsForm	Відкрито головне вікно, завантажено список полів користувача.	Успішно
2	Авторизація з невірними даними	Логін: admin	Блокування входу, виведення повідомлення:	Виведено вікно з помилкою, вхід	Успішно
3	Додавання нового поля	Назва: Поле №1 Площа: 100 Кількість секцій: 4 Клімат:Помірний	Створення запису в таблиці fields, генерація 4-х секцій в таблиці section, оновлення	Поле відобразилося в списку, секції успішно створено.	Успішно
4	Виконання агрозаходу (Полив)	Секція: 1 Об'єм: 50 Ціна: 15	Запис у таблицю AgroOperations, збільшення витрат (expenses) на 500, збільшення	Витрати зросли на 500 грн, волога підвищилася.	Успішно
5	Запуск щотижневої симуляції	Дата: 28.05.2026 Температура: 22 Опади: 5	Розрахунок за моделлю: приріст висоти (GDD), зміна NPK (Лібіх). Запис нового рядка в WeeklyMeasurements.	Новий запис успішно додано, графіки інтерфейсу оновлено.	Успішно

Рис. 4.1 Результати функціонального тестування інформаційної системи

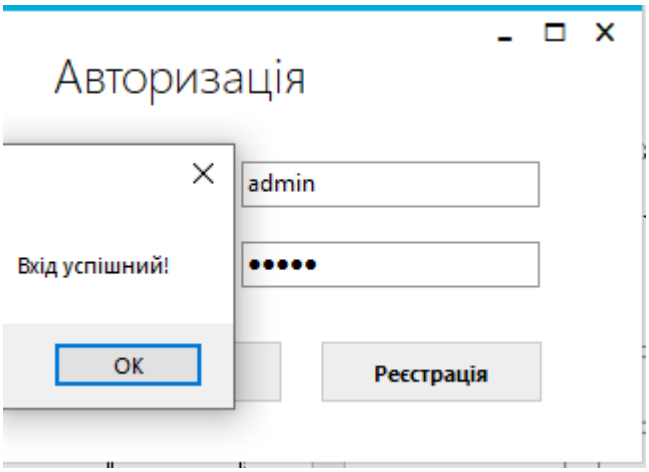


Рис. 4.2 Успішна авторизація

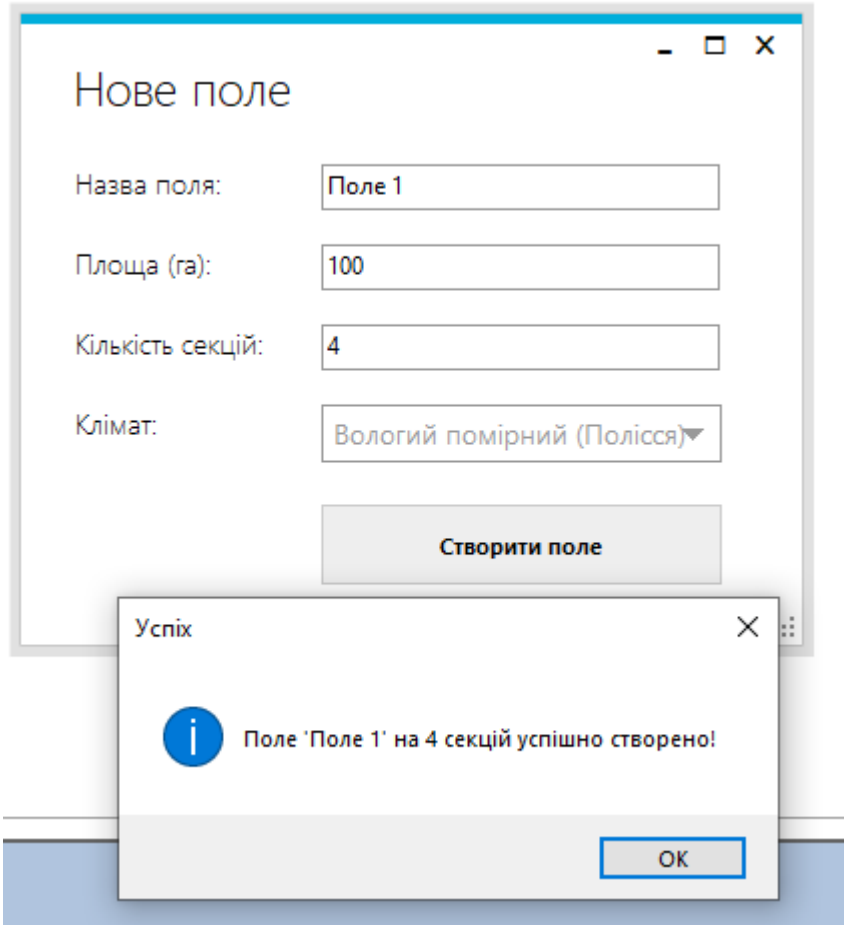


Рис. 4.3 Створення поля

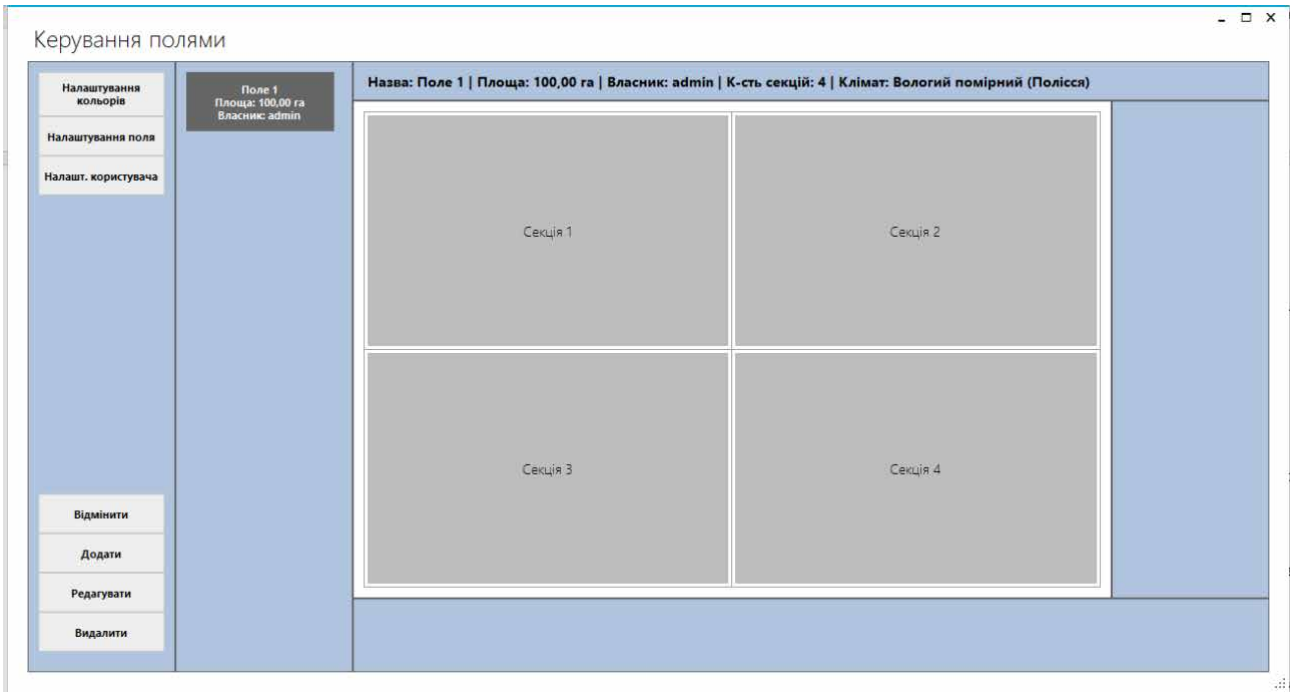


Рис. 4.4 Загальний вигляд

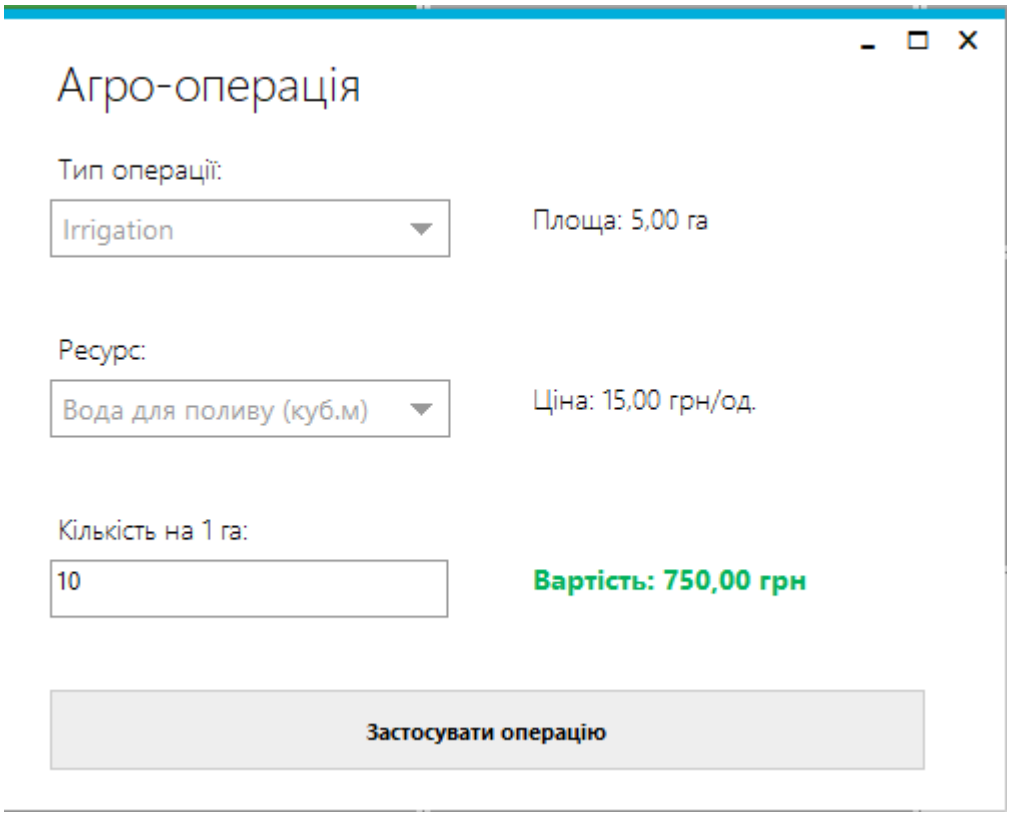


Рис. 4.5 Проведення операції поливу

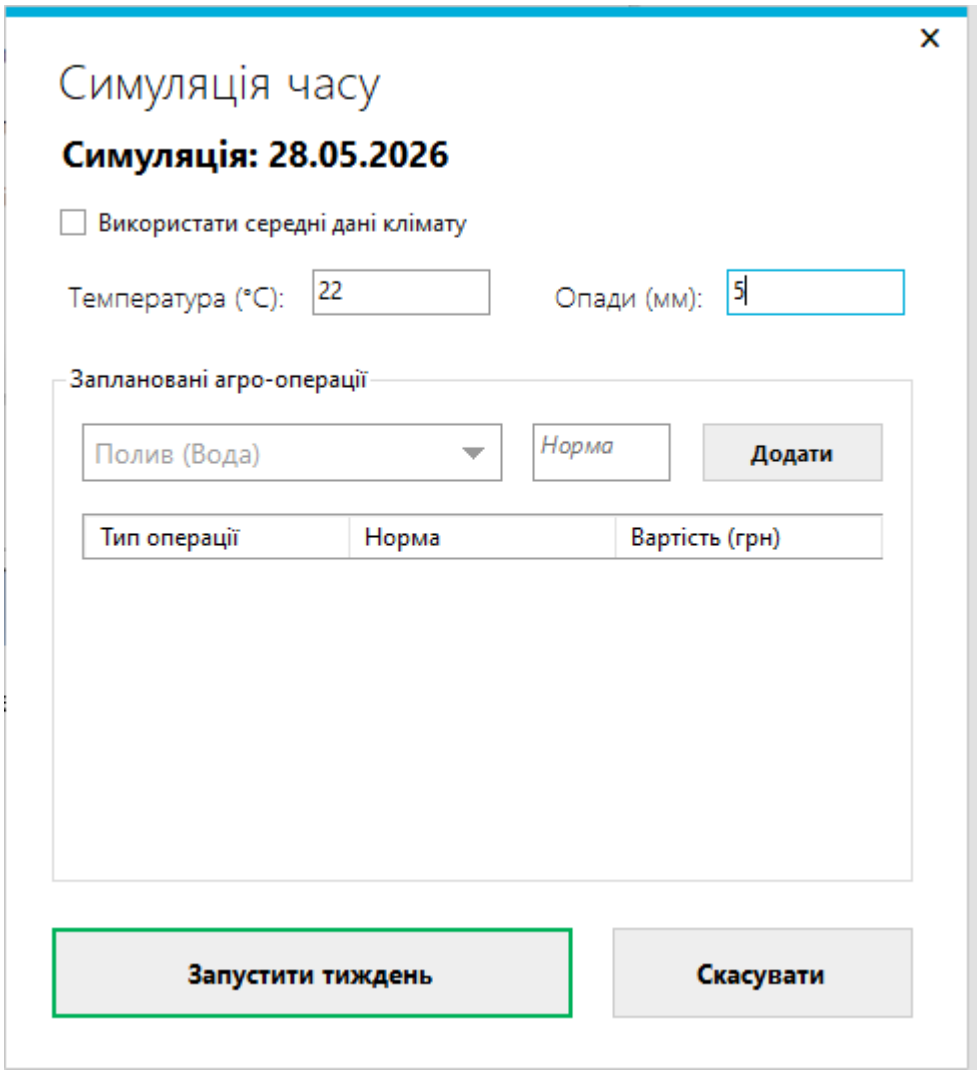


Рис. 4.6 Проведення симуляції

4.2 Вимоги до апаратного та програмного забезпечення

Для визначення вимог до обладнання необхідно спроєктувати фізичну архітектуру розгортання розробленої програмної системи. З цією метою побудовано діаграму розміщення (топології) UML (Deployment Diagram), яка визначає апаратні вузли системи та місцезнаходження програмних компонентів на них (Рис. 4.7).

Відповідно до представленої топології, система може працювати у двох режимах: ізольованому (коли клієнтський і серверний вузли знаходяться на одному фізичному комп'ютері) та мережевому клієнт-серверному (коли база даних розгорнута на виділеному сервері підприємства).

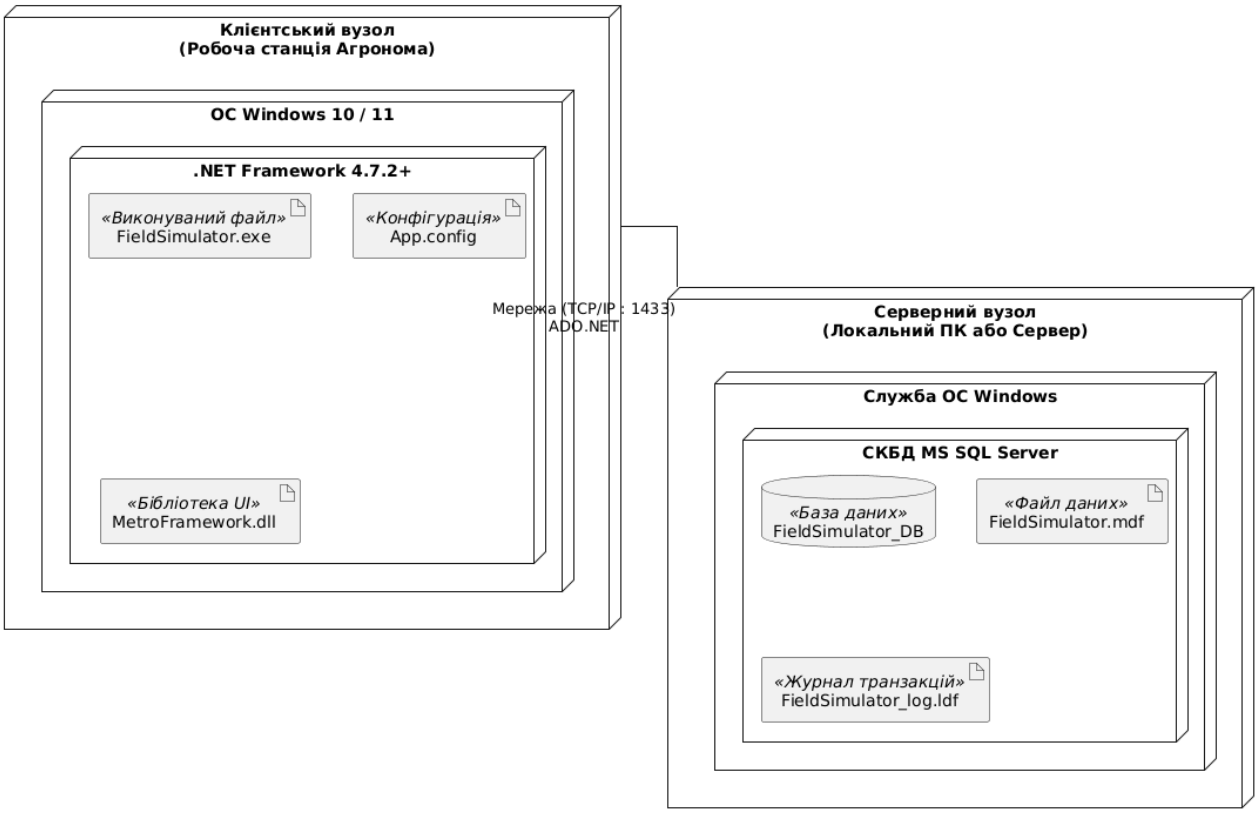


Рис. 4.7 Діаграма розгортання системи

Виходячи з обраної архітектури ПЗ, були визначені наступні вимоги до апаратного забезпечення клієнтської робочої станції:

- Процесор (CPU): двоядерний процесор з тактовою частотою від 2.0 ГГц (наприклад, Intel Core i3 або аналог від AMD).
- Оперативна пам'ять (RAM): мінімум 4 ГБ (рекомендовано 8 ГБ для комфортної роботи СКБД у локальному режимі).
- Дисковий простір: мінімум 500 МБ для встановлення програми та початкових файлів бази даних (рекомендується SSD-накопичувач для швидкого виконання транзакцій).
- Монітор: з роздільною здатністю не менше 1366x768 пікселів (рекомендовано 1920x1080 для зручного перегляду таблиць).

Вимоги до програмного (системного) забезпечення:

- Операційна система: Microsoft Windows 10 або Windows 11 (64-bit).
- Платформа виконання: Microsoft .NET Framework версії 4.7.2 або новішої.

- Система управління базами даних: MS SQL Server 2019 Express (або новіша версія) з увімкненою підтримкою протоколу TCP/IP або локальних підключень (Named Pipes).

4.3 Склад інсталяційного пакету

Для того, щоб запрацювала розроблена інформаційна система «FieldSimulator», її необхідно коректно інсталювати на цільовому обладнанні. Оскільки система складається з прикладної програми та реляційної бази даних, процес встановлення передбачає розгортання сховища даних на локальному сервері СКБД та налаштування клієнтського додатка.

Склад інсталяційного пакету чітко структурований. Інсталяційний пакет (у вигляді ZIP-архіву або директорії розгортання) містить такі компоненти:

1. **Виконуваний модуль програми:**
    - FieldSimulator.exe - головний скомпільований файл запуску десктопного додатка.
2. **Конфігураційний файл налаштувань:**
    - FieldSimulator.exe.config (генерується з App.config) - XML-файл, що містить глобальні параметри додатку. У цьому файлі винесено рядок підключення (ConnectionString). Під час встановлення системи адміністратор або користувач має відкрити цей файл у текстовому редакторі та змінити параметр Data Source на актуальне ім'я свого локального або мережевого сервера MS SQL (наприклад, .\SQLEXPRESS). Це дозволяє розгортати систему без необхідності перекомпіляції вихідного коду.
3. **Бібліотеки залежностей (DLL):**
    - MetroFramework.dll, MetroFramework.Design.dll, MetroFramework.Fonts.dll - набір бібліотек, необхідних для коректного відтворення сучасного графічного інтерфейсу (Flat UI).

4. **Скрипти розгортання бази даних (Каталог Database_Scripts):** Для забезпечення чистого та безпомилкового розгортання, ініціалізацію бази даних розділено на два послідовні етапи за допомогою SQL-скриптів:
    - 01_CreateSchema.sql (Data Definition Language) - скрипт, який відповідає виключно за створення структури бази даних: створення 13 таблиць, визначення первинних і зовнішніх ключів, типів даних та обмежень цілісності (Constraints).
    - 02_InsertInitialData.sql (Data Manipulation Language) - скрипт, який запускається другим і наповнює системні таблиці-довідники початковими базовими даними, необхідними для роботи програми (номенклатура сільськогосподарських культур, базові показники кліматичних зон, перелік агроресурсів та дефолтний обліковий запис адміністратора).
5. **Документація:**
    - Readme.txt або Install_Manual.pdf - покрокова інструкція для користувача щодо виконання SQL-скриптів у середовищі SQL Server Management Studio (SSMS) та налаштування файлу конфігурації перед першим запуском програми.

Запропонований склад інсталяційного пакету є самодостатнім, гнучким до зміни апаратного середовища та гарантує повну працездатність програмного продукту після виконання інструкцій з розгортання.

Для максимального спрощення процесу розгортання системи кінцевим користувачем, усі вищезазначені компоненти (виконуваний файл, конфігурації, бібліотеки MetroFramework та SQL-скрипти) були запаковані у єдиний виконуваний інсталяційний файл (наприклад, FieldSimulator_Setup.exe). Збірка інсталятора здійснюється за допомогою професійного компілятора **Inno Setup**. Використання Inno Setup дозволяє автоматизувати процес встановлення: програма-інсталятор самостійно створює необхідну ієрархію директорій у системній папці Program Files, копіює туди файли залежностей, створює ярлики для запуску на робочому столі користувача та надає зручний інтерфейс

деінсталяції програми. Каталог зі скриптами бази даних розпаковується у підпапку Database_Scripts поруч із головним файлом, що дозволяє адміністратору швидко знайти їх для ініціалізації БД.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено актуальне науково-прикладне завдання - спроєктовано та розроблено програмне забезпечення інформаційної системи «FieldSimulator», призначеної для відслідковування динаміки розвитку посівів сільськогосподарських культур та економічного обліку агрозаходів.

Виконання роботи дозволило дійти наступних висновків:

1. На етапі системного аналізу предметної області визначено ключові проблеми планування та моніторингу в аграрному секторі. Сформульовано чіткі функціональні та нефункціональні вимоги до програмного продукту. Побудовано концептуальну модель системи, визначено основні прецеденти (Use Cases) та ролі користувачів.
2. Проведено проєктування архітектури та інформаційного забезпечення. Розроблено логічну та фізичну моделі реляційної бази даних, нормалізовані до Третьої нормальної форми (3НФ), що унеможливлює виникнення аномалій даних. За допомогою об'єктно-орієнтованого підходу та діаграм UML (класів, пакетів, компонентів) побудовано класичну трирівневу архітектуру системи (3-Tier Architecture), яка забезпечила низьку зв'язність інтерфейсу з механізмами доступу до даних.
3. Програмну реалізацію системи виконано мовою C# на платформі .NET Framework з використанням компонентів MetroFramework для побудови сучасного ергономічного інтерфейсу (Flat UI). Інтеграцію з СУБД MS SQL Server реалізовано за допомогою технології ADO.NET із застосуванням патерну Singleton для оптимізації пулу підключень.
4. Розроблено та інтегровано в програмний код алгоритми математичного моделювання розвитку посівів. Для забезпечення наукової достовірності розрахунків застосовано загальноприйняті агрономічні моделі: оцінка водного балансу з урахуванням евапотранспірації (на базі підходів FAO-

56), розрахунок росту біомаси за сумою активних температур (GDD) та визначення стрес-факторів і споживання макроелементів (NPK) за Законом мінімуму Лібіха.

5. Реалізовано транзакційний механізм економічного обліку, який дозволяє в рамках єдиної атомарної операції фіксувати проведення агрозаходу (полив, удобрення), змінювати параметри вологи/нутрієнтів у базі даних та динамічно перераховувати фінансові витрати поля на основі кастомних цін користувача.
6. Проведено функціональне тестування системи за методом «чорного ящика», результати якого підтвердили її працездатність, надійність та готовність до експлуатації. Процес розгортання автоматизовано шляхом створення комплексного інсталяційного пакету за допомогою утиліти Inno Setup та підготовки скриптів ініціалізації бази даних.

Таким чином, розроблене програмне забезпечення повністю задовольняє поставлені вимоги, має високу практичну цінність і може бути впроваджене в діяльність малих та середніх агропромислових підприємств для підвищення ефективності управління земельними ресурсами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Троелсен Е., Джепікс Ф. Мова програмування C# 7.0 та платформа .NET 4.7. 8-ме вид. Київ: Діалектика, 2018. 1328 с.
2. Макконнелл С. Досконалий код. Майстер-клас. Київ: КМ-Букс, 2019. 896 с.
3. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Патерни об’єктно-орієнтованого проєктування. Харків: Фабула, 2020. 416 с.
4. Конноллі Т., Бегг К. Бази даних. Проєктування, реалізація і супровід. Теорія і практика. Київ: Діалектика, 2018. 1440 с.
5. Фаулер М. Архітектура корпоративних програмних додатків. Київ: Кліо, 2019. 544 с.
6. Буч Г., Рамбо Д., Джекобсон І. UML. Посібник користувача. 2-ге вид. Київ: ДІАВЕК, 2018. 496 с.
7. Мартін Р. Чистий код. Створення і рефакторинг за допомогою Agile. Харків: Фабула, 2019. 416 с.
8. Фрімен Е., Сьєрра К., Бейтс Б. Патерни проєктування Head First. Харків: Фабула, 2020. 672 с.
9. Соммервілл І. Інженерія програмного забезпечення. Київ: ВД "Києво-Могилянська академія", 2018. 528 с.
10. Клеппман М. Обробка даних: проєктування надійних систем. Київ: Форс Україна, 2021. 608 с.
11. Allen R. G., Pereira L. S., Raes D., Smith M. Crop evapotranspiration - Guidelines for computing crop water requirements. FAO Irrigation and drainage paper 56. Rome: Food and Agriculture Organization of the United Nations, 1998. 300 p.
12. Майерс Г. Мистецтво тестування програмного забезпечення. Київ: Діалектика, 2018. 272 с.

13. Офіційна документація щодо мови C# та платформи .NET. Microsoft : веб-сайт. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 10.05.2026).
14. Офіційна документація по архітектурі ADO.NET. Microsoft : веб-сайт. URL: https://learn.microsoft.com/uk-ua/dotnet/framework/data/adonet/ (дата звернення: 12.05.2026).
15. Документація користувача Microsoft SQL Server. Microsoft : веб-сайт. URL: https://learn.microsoft.com/uk-ua/sql/sql-server/ (дата звернення: 14.05.2026).
16. PlantUML: Open-source tool for drawing UML diagrams. URL: https://plantuml.com/ (дата звернення: 15.05.2026).
17. MetroFramework UI Modern UI for Windows Forms. GitHub : веб-сайт. URL: https://github.com/thielj/MetroFramework (дата звернення: 16.05.2026).
18. Inno Setup Compiler Documentation. JRSoftware : веб-сайт. URL: https://jrsoftware.org/isinfo.php (дата звернення: 17.05.2026).
19. Liebig's law of the minimum. Biology Dictionary : веб-сайт. URL: https://biologydictionary.net/liebigs-law-of-the-minimum/ (дата звернення: 18.05.2026).
20. Growing Degree Days (GDD) and Crop Development. Cornell University : веб-сайт. URL: https://climatesmartfarming.org/tools/csf-growing-degree-day-calculator/ (дата звернення: 19.05.2026).
21. SOLID Principles of Object-Oriented Design. DigitalOcean : веб-сайт. URL: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (дата звернення: 20.05.2026).
22. Singleton Design Pattern in C#. DotNetTricks : веб-сайт. URL: https://www.dotnettricks.com/learn/designpatterns/singleton-design-pattern-c-sharp (дата звернення: 20.05.2026).
23. Web-технології та Web-дизайн: Застосування мови HTML для створення електронних ресурсів // Навчальний посібник. (видання друге) / – Київ: Видавництво Ліра-К, 2020. - 212 с.

24. Бородкіна І.Л., Бородкін Г.О. Теорія алгоритмів: Навчальний посібник. – Київ : Центр учбової літератури, 2018. – 184 с.
25. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення: Навчальний посібник. – Київ : Центр учбової літератури, 2018. - 204 с.

ДОДАТКИ

ДОДАТОК А

Код створення таблиць

Сторінок 6

```
-- ================================================================
-- 1. СТВОРЕННЯ БАЗИ ДАНИХ
-- ================================================================
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'FieldSimulatorDB')
BEGIN
   CREATE DATABASE FieldSimulatorDB;
END
GO

USE FieldSimulatorDB;
GO

-- ================================================================
-- 2. ВИДАЛЕННЯ ТАБЛИЦЬ (З урахуванням ієрархії зовнішніх ключів)
-- ================================================================
IF OBJECT_ID('WeeklyMeasurements', 'U') IS NOT NULL DROP TABLE WeeklyMeasurements;
IF OBJECT_ID('AgroOperations', 'U') IS NOT NULL DROP TABLE AgroOperations;
IF OBJECT_ID('section', 'U') IS NOT NULL DROP TABLE section;
IF OBJECT_ID('fields', 'U') IS NOT NULL DROP TABLE fields;
IF OBJECT_ID('UserCropPrices', 'U') IS NOT NULL DROP TABLE UserCropPrices;
IF OBJECT_ID('UserResourcePrices', 'U') IS NOT NULL DROP TABLE UserResourcePrices;
IF OBJECT_ID('UserSettings', 'U') IS NOT NULL DROP TABLE UserSettings;
IF OBJECT_ID('ColorSettings', 'U') IS NOT NULL DROP TABLE ColorSettings;
IF OBJECT_ID('field_backup', 'U') IS NOT NULL DROP TABLE field_backup;
IF OBJECT_ID('users', 'U') IS NOT NULL DROP TABLE users;
IF OBJECT_ID('crops', 'U') IS NOT NULL DROP TABLE crops;
IF OBJECT_ID('Resources', 'U') IS NOT NULL DROP TABLE Resources;
IF OBJECT_ID('Climates', 'U') IS NOT NULL DROP TABLE Climates;
GO

-- ================================================================
```

```
-- 3. СТВОРЕННЯ БАЗОВИХ ТАБЛИЦЬ (Батьківські таблиці)
-- ======================================================================

-- Таблиця користувачів
CREATE TABLE users (
   u_id INT IDENTITY(1,1) PRIMARY KEY,
   login NVARCHAR(50) NOT NULL UNIQUE,
   password NVARCHAR(255) NOT NULL
);

-- Довідник культур (включає всі обговорені агрономічні колонки)
CREATE TABLE crops (
   crop_id INT IDENTITY(1,1) PRIMARY KEY,
   crop_name NVARCHAR(100) NOT NULL,
   crop_tOpt DECIMAL(5,2),
   crop_wOpt DECIMAL(5,2),
   crop_pOpt DECIMAL(5,2),
   crop_hOpt DECIMAL(5,2),
   crop_CCI DECIMAL(5,2) DEFAULT 1.0,
   crop_WCI DECIMAL(5,2) DEFAULT 1.0,
   crop_NBI DECIMAL(5,2) DEFAULT 1.0,
   crop_GPI DECIMAL(5,2) DEFAULT 1.0,
   defAzot DECIMAL(8,2),
   defPhosfor DECIMAL(8,2),
   defCaliy DECIMAL(8,2),
   default_planting_cost_per_ha DECIMAL(18,2),
   default_harvesting_cost_per_ha DECIMAL(18,2),
   default_selling_price_per_ton DECIMAL(18,2),
   base_yield_ha DECIMAL(8,2),

);
```

```
-- Довідник ресурсів (вода, добрива, паливо)
CREATE TABLE Resources (
   resource_id INT IDENTITY(1,1) PRIMARY KEY,
   resource_name NVARCHAR(100) NOT NULL,
   unit NVARCHAR(20),
   default_price_per_unit DECIMAL(18,2),
   category NVARCHAR(50)
);

-- Довідник кліматичних зон
CREATE TABLE Climates (
   climate_id INT IDENTITY(1,1) PRIMARY KEY,
   climate_name NVARCHAR(100) NOT NULL,
   spring_temp DECIMAL(5,2),
   spring_precip DECIMAL(5,2),
   summer_temp DECIMAL(5,2),
   summer_precip DECIMAL(5,2),
   autumn_temp DECIMAL(5,2),
   autumn_precip DECIMAL(5,2),
   winter_temp DECIMAL(5,2),
   winter_precip DECIMAL(5,2)
);

-- Таблиця для бекапів полів
CREATE TABLE field_backup (
   field_id INT PRIMARY KEY,
   field_name NVARCHAR(100),
   field_area DECIMAL(18,2),
   field_sections INT
);
```

```
-- ================================================================
-- 4. СТВОРЕННЯ ЗАЛЕЖНИХ ТАБЛИЦЬ (Налаштування користувача)
-- ================================================================

CREATE TABLE UserSettings (
  setting_id INT IDENTITY(1,1) PRIMARY KEY,
  u_id INT NOT NULL FOREIGN KEY REFERENCES users(u_id),
  enable_smart_prompts BIT DEFAULT 1
);

CREATE TABLE ColorSettings (
  setting_id INT IDENTITY(1,1) PRIMARY KEY,
  u_id INT NOT NULL FOREIGN KEY REFERENCES users(u_id),
  status_name NVARCHAR(50),
  color_hex NVARCHAR(10)
);

CREATE TABLE UserCropPrices (
  ucp_id INT IDENTITY(1,1) PRIMARY KEY,
  u_id INT NOT NULL FOREIGN KEY REFERENCES users(u_id),
  crop_id INT NOT NULL FOREIGN KEY REFERENCES crops(crop_id),
  custom_planting_cost DECIMAL(18,2),
  custom_harvesting_cost DECIMAL(18,2),
  custom_selling_price DECIMAL(18,2)
);

CREATE TABLE UserResourcePrices (
  urp_id INT IDENTITY(1,1) PRIMARY KEY,
  u_id INT NOT NULL FOREIGN KEY REFERENCES users(u_id),
  resource_id INT NOT NULL FOREIGN KEY REFERENCES Resources(resource_id),
  custom_price DECIMAL(18,2)
```

```
);
-- ================================================================
-- 5. СТВОРЕННЯ ОПЕРАЦІЙНИХ ТАБЛИЦЬ (Поля, Секції, Операції)
-- ================================================================

-- Поля
CREATE TABLE fields (
   field_id INT IDENTITY(1,1) PRIMARY KEY,
   u_id INT NOT NULL FOREIGN KEY REFERENCES users(u_id),
   climate_id INT NOT NULL FOREIGN KEY REFERENCES Climates(climate_id),
   field_name NVARCHAR(100) NOT NULL,
   field_area DECIMAL(18,2) NOT NULL,
   field_sections INT NOT NULL,
   inherit_user_settings BIT DEFAULT 1,
   local_planting_cost DECIMAL(18,2),
   local_harvesting_cost DECIMAL(18,2)
);

-- Секції поля (конкретні ділянки під посів)
CREATE TABLE section (
   section_id INT IDENTITY(1,1) PRIMARY KEY,
   field_id INT NOT NULL FOREIGN KEY REFERENCES fields(field_id),
   section_number INT NOT NULL,
   crop_id INT NULL FOREIGN KEY REFERENCES crops(crop_id),
   planting_date DATETIME NULL,
   section_index_AEI DECIMAL(5,4) DEFAULT 1.0,
   NBI DECIMAL(5,4) DEFAULT 1.0,
   GPI DECIMAL(5,4) DEFAULT 1.0,
   CCI DECIMAL(5,4) DEFAULT 1.0,
   WCI DECIMAL(5,4) DEFAULT 1.0,
   expenses DECIMAL(18,2) DEFAULT 0,
```

```
  expected_revenue DECIMAL(18,2) DEFAULT 0,
    profit DECIMAL(18,2) DEFAULT 0
);

-- Агротехнічні операції (полив, удобрення)
CREATE TABLE AgroOperations (
    operation_id INT IDENTITY(1,1) PRIMARY KEY,
    section_id INT NOT NULL FOREIGN KEY REFERENCES section(section_id),
    resource_id INT NOT NULL FOREIGN KEY REFERENCES Resources(resource_id),
    operation_type NVARCHAR(50),
    operation_date DATETIME NOT NULL,
    quantity_used DECIMAL(18,2) NOT NULL
);

-- Щотижневі вимірювання (моніторинг стану)
CREATE TABLE WeeklyMeasurements (
    monitoring_id INT IDENTITY(1,1) PRIMARY KEY,
    section_id INT NOT NULL FOREIGN KEY REFERENCES section(section_id),
    measurement_date DATETIME NOT NULL,
    Height DECIMAL(8,2),
    Temp DECIMAL(5,2),
    Precipitation DECIMAL(8,2),
    Humidity DECIMAL(5,2),
    IrrigationsCount INT DEFAULT 0,
    Azot DECIMAL(8,2),
    Phosfor DECIMAL(8,2),
    Caliy DECIMAL(8,2)
);
GO
```

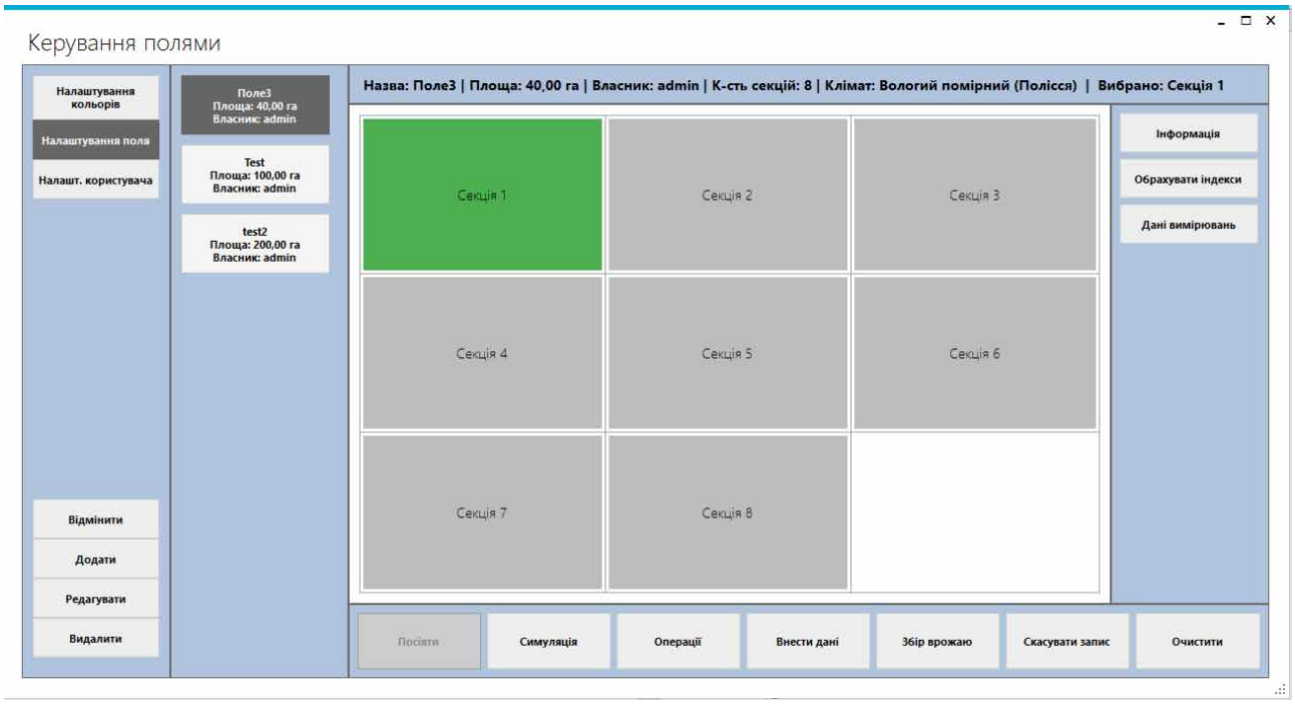
ДОДАТОК Б

Скріншоти робої програми

Сторінок 3

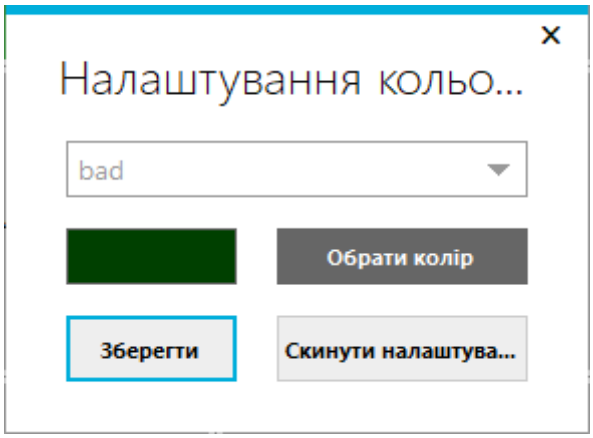
Сторінка 1

Основна сторінка з вибраною не засадженою секцією

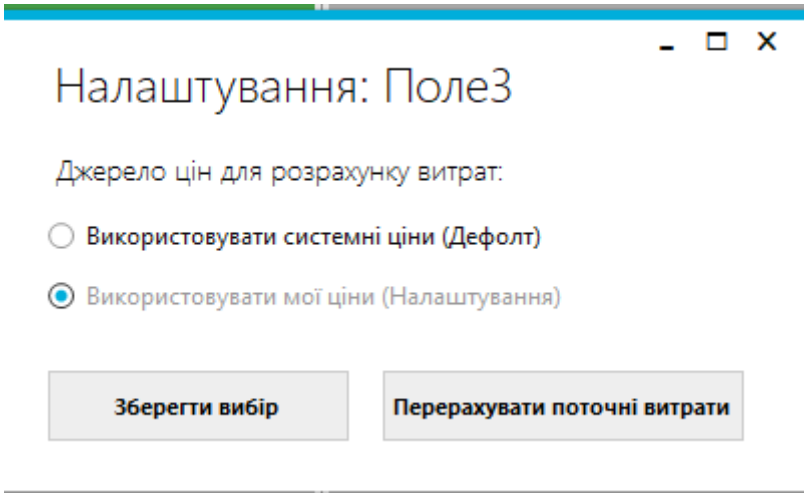


Основна сторінка із вибраною секцією

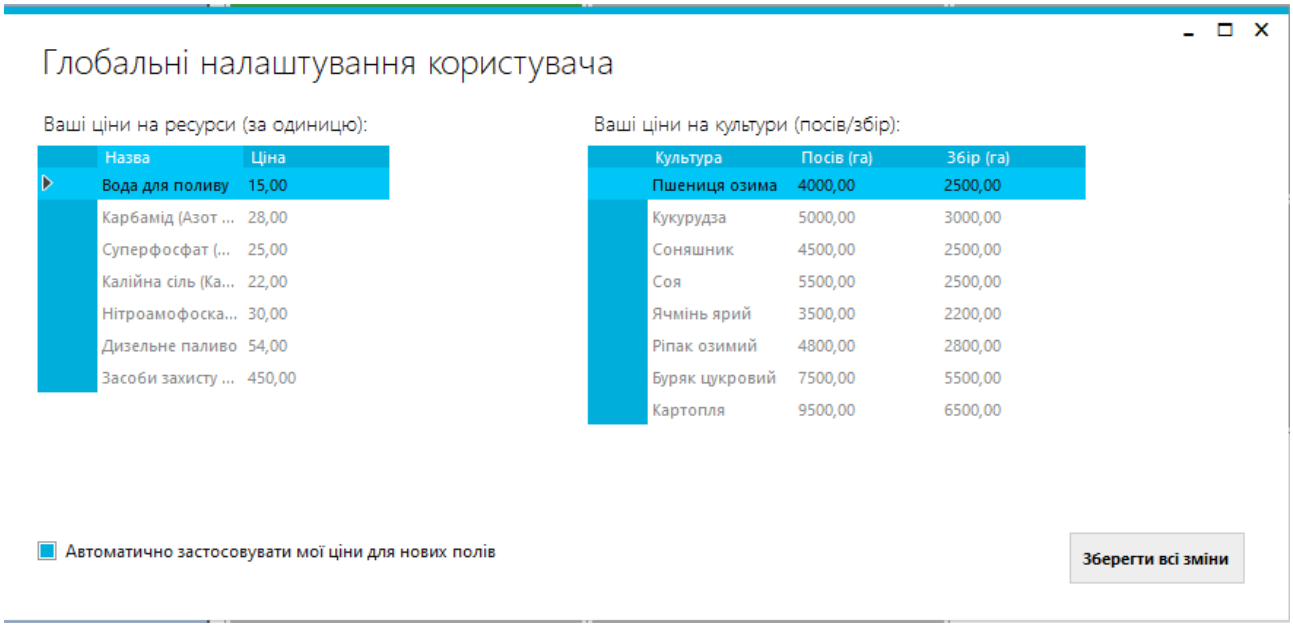
Скріншоти налаштувань



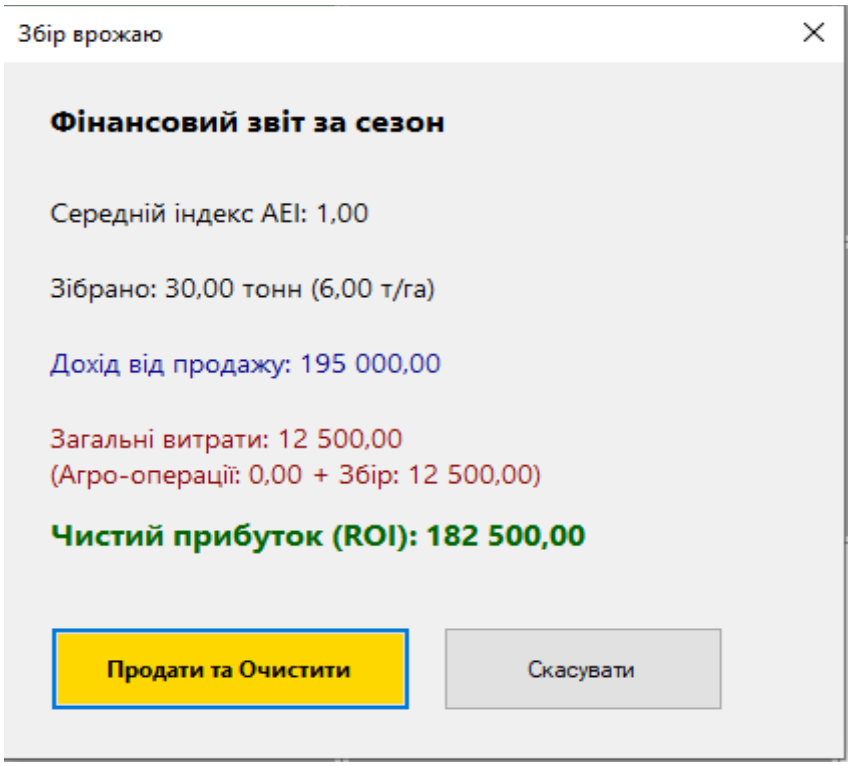
Кольори



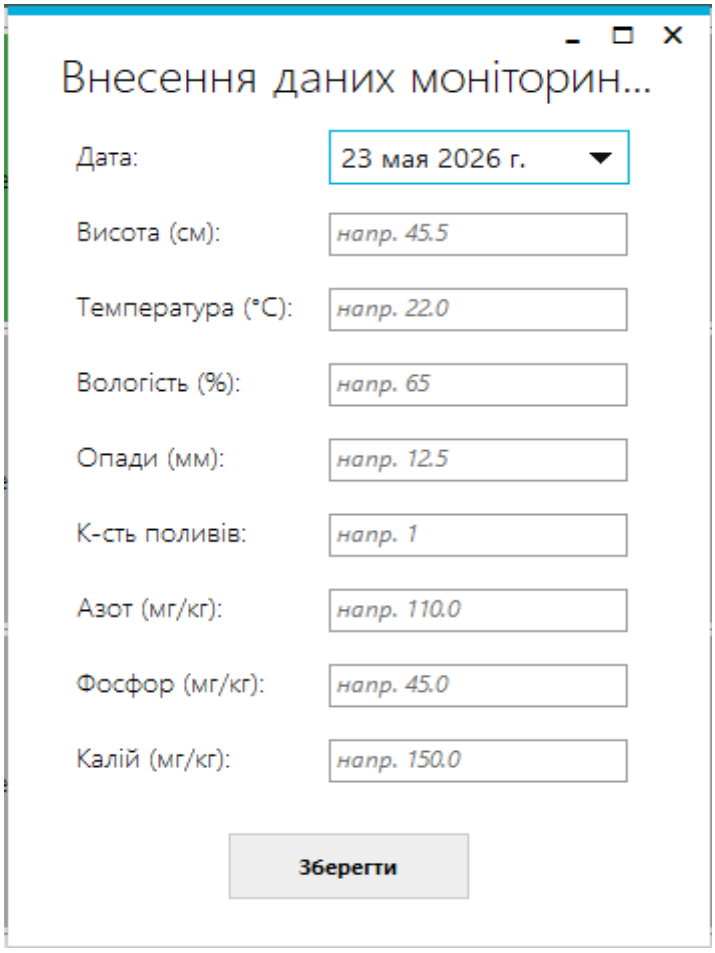
Налаштування вибраного поля



Налаштування користувача



Фінансовий звіт



Внесення показників

ДОДАТОК В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Бортніков Дмитро Сергійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення системи відслідковування динаміки розвитку сільськогосподарських культур» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [ ] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;
        - [ ] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Дмитро БОРТНІКОВ**
(підпис)