НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
__інформаційних систем і технологій_
(назва кафедри)

_____ / Швиденко М. З. /
(підпис) (ПІБ)

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна система проведення лабораторного сортового контролю»

Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми
_____д. ек. н, професор_____ _____ _____ Руденський Р.А_____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____ к. ек. н._____ _____ __Стариченко Є. М._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Вовк Денис Олександрович_____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
__інформаційних систем і технологій__
(назва кафедри)

___к.ек. н, доцент _____ Швиденко М. З.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

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

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система проведення лабораторного сортового контролю» затверджена наказом ректора НУБіП України від 16.12.2024 № 2246 "С"
2. Термін подання завершеної роботи на кафедру _____02.06.2025 р._____
(рік, місяць, число)
3. Вихідні дані до роботи: надання інформації про результати лабораторного дослідження у вигляді звітів та графіків про проби насіння.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "16" грудня 2024 р.

Керівник бакалаврської кваліфікаційної роботи _____ / Стариченко Є.М. /
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____ / Вовк Д.О. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У бакалаврській роботі представлено розробку веб-системи, призначеної для автоматизації та оптимізації процесу лабораторного сортового контролю. Сортовий контроль є критично важливим компонентом у забезпеченні якості сільськогосподарської продукції, гарантуючи відповідність зразків насіння заявленим сортовим характеристикам перед їх використанням у виробництві чи розповсюдженні. Запропонована система полегшує повний цикл сортового контролю, включаючи створення та керування записами зразків насіння, формування договорів на відправку зразків до лабораторій, а також введення та формування результатів випробувань.

Додаток має на меті замінити традиційні паперові процеси цифровою платформою, покращуючи ефективність, відстежуваність і точність даних. Для реалізації системи використано мови програмування PHP, JavaScript, СУБД MySQL, інструменти Symfony, Doctrine ORM та сучасні фреймворки для створення графічного інтерфейсу користувача.

Результатом цієї роботи є функціональний прототип інформаційної системи, яка може бути розгорнута в лабораторних умовах для підтримки робочих процесів сортовипробування. Система демонструє потенціал підвищення продуктивності та зменшення помилок при обробці даних, одночасно підвищуючи загальну надійність процедур контролю сортів.

ABSTRACT

This bachelor's thesis presents the development of a web-based system designed to automate and optimize the process of laboratory variety control. Variety control is a critical component in ensuring the quality of agricultural products, guaranteeing that seed samples comply with the declared varietal characteristics before their use in production or distribution. The proposed system facilitates the full cycle of variety control, including the creation and management of seed sample records, the formation of contracts for sending samples to laboratories, as well as the entry and formation of test results.

The application aims to replace traditional paper processes with a digital platform, improving efficiency, traceability and data accuracy. To implement the system, the programming languages PHP, JavaScript, MySQL DBMS, Symfony, Doctrine ORM tools, and modern frameworks for creating a graphical user interface were used.

The result of this work is a functional prototype of an information system that can be deployed in laboratory conditions to support variety testing workflows. The system demonstrates the potential to increase productivity and reduce errors in data processing, while increasing the overall reliability of variety control procedures.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	16 грудня 2024	
2	Аналіз предметної області	березень 2025	
3	Моделювання предметної області	березень 2025	
4	Проєктування програмної системи	квітень 2025	
5	Розгортання та експлуатація програмного забезпечення	квітень- травень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	квітень- травень 2025	
7	Оформлення записки	травень 2025	
8	Перевірка на плагіат	травень 2025	
9	Проходження нормо контроля	травень 2025	
10	Проходження передзахисту	2 червня 2025	
11	Захист роботи	червень 2025	

Студент _____ Вовк Д.О.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.ек.н., доцент _____ Стариченко Є.М.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

У сучасному сільському господарстві якість і автентичність садивного матеріалу відіграють ключову роль у забезпеченні високих врожаїв і сталого виробництва продуктів харчування. Одним із найважливіших аспектів оцінки якості насіння є сортовий контроль, який визначає відповідність зразків насіння заявленим сортовим ознакам. Цей процес зазвичай виконується в спеціалізованих лабораторіях, які дотримуються суворих протоколів і стандартів.

Актуальність цієї роботи полягає у зростаючому попиті на ефективні та надійні інструменти для підтримки забезпечення якості в сільському господарстві, зокрема, у сфері контролю за сортами насіння. Оскільки лабораторії стикаються зі збільшенням обсягів зразків насіння та суворішими нормативними вимогами, традиційних паперових або напівцифрових методів уже недостатньо для забезпечення своєчасної та точної обробки даних. Розробка інформаційної системи для керування зразками насіння, контрактами та результатами випробувань вирішує ці проблеми шляхом автоматизації рутинних завдань, мінімізації помилок людини та покращення відстеження. Така система не тільки покращує продуктивність лабораторії, але й сприяє загальній прозорості та надійності процесів сертифікації насіння, які життєво важливі для сталого розвитку сільського господарства та довіри ринку.

Незважаючи на важливість сортового контролю, багато лабораторій все ще покладаються на паперові або фрагментовані цифрові методи для керування зразками насіння, документування контрактів і реєстрації результатів випробувань. Ці застарілі підходи забирають багато часу, схильні до людських помилок і перешкоджають доступності та відстеженню даних. У відповідь на ці виклики зростає потреба в централізованому автоматизованому

рішенні, яке могло б оптимізувати лабораторні робочі процеси та забезпечити надійне керування даними.

Ця бакалаврська робота розглядає проблему шляхом розробки веб-інформаційної системи для проведення лабораторного контролю сортів. Система призначена для підтримки повного циклу сортового контролю, включаючи:

- створення та керування записами зразків насіння;
- оформлення договорів на подачу зразків насіння на лабораторні дослідження;
- введення та формування результатів тестування в цифровому форматі;
- зберігання та отримання історичних записів для подальшого використання та звітності.

Метою цієї дипломної роботи є привнесення у галузь сільського господарства шляху для полегшення проведення перевірки проб насіння, розробивши програмне рішення, яке покращує ефективність, прозорість та точність лабораторних процесів, пов’язаних із контролем сортів. Система розроблена з використанням сучасних веб-технологій з акцентом на зручності використання, цілісності даних і масштабованості.

Ця робота включає аналіз предметної області, визначення системних вимог, проєктування її архітектури та реалізацію робочого прототипу. Отриманий додаток надасть лабораторіям і сільськогосподарським установам практичний інструмент для управління сортовипробуванням і підтримки прийняття обґрунтованих рішень щодо сертифікації та розповсюдження насіння.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сортовий контроль – важливий процес в аграрному секторі, спрямований на перевірку відповідності зразків насіння заявленим сортовим ознакам. Цей тип контролю гарантує, що насіння, яке використовується в сільськогосподарському виробництві, належить до правильного сорту, що є життєво важливим для збереження якості врожаю, сталості врожайності та генетичної чистоти. Процес зазвичай проводиться в спеціалізованих лабораторіях, які дотримуються встановлених процедур і правил, часто відповідно до національних або міжнародних схем сертифікації насіння.

У поточній практиці багато лабораторій все ще покладаються на ручні методи або різні цифрові інструменти для управління різними етапами сортового контролю. Це включає реєстрацію зразків насіння, складання договорів із виробниками або постачальниками насіння, відстеження статусу випробувань і створення остаточних звітів на основі лабораторних спостережень. Такі фрагментовані робочі процеси схильні до неефективності, включаючи повторюване введення даних, труднощі з отриманням історичних записів, відсутність відстеження в реальному часі та підвищений ризик помилок.

Крім того, відсутність централізованої інформаційної системи ускладнює забезпечення прозорості, цілісності даних і можливості відстеження — ключових вимог регуляторних перевірок і аудитів. Лабораторії, які здійснюють контроль сортів, повинні дотримуватися суворих стандартів документації та звітності, яких важко виконати без структурованого автоматизованого підходу до управління даними.

Існує очевидна потреба в веб-рішенні, яке об’єднує всі етапи процесу сортового контролю в єдину платформу. Це оптимізує операції, забезпечить швидший доступ до інформації, підтримає кращий зв’язок між залученими

сторонами (такими як лабораторії, виробники насіння та регуляторні органи) і зменшить адміністративне навантаження на персонал лабораторії. Розробка такої системи не тільки підвищить ефективність сортоконтролю, але й підвищить загальну якість та довіру до процесів сертифікації насіння в сільськогосподарській галузі.

Лабораторний сортовий контроль є критично важливим процесом у сучасній сільськогосподарській практиці, призначений для перевірки автентичності та чистоти сортів насіння перед їх впровадженням у великомасштабне виробництво. Ця перевірка має важливе значення для забезпечення стабільності таких характеристик культури, як потенціал врожайності, стійкість до хвороб і адаптованість до конкретних умов середовища. Результат цього процесу безпосередньо впливає на продуктивність сільського господарства та якість ланцюга постачання продовольства [3].

У професійних лабораторіях сортовий контроль включає послідовність операцій, починаючи з прийому та ідентифікації зразків насіння, поданих виробниками або контролюючими органами. Кожен зразок має бути належним чином задокументований, часто включаючи такі деталі, як заявлений сорт, походження та номер партії. Після реєстрації лабораторія ініціює організаційну та правову основу для тестування, як правило, шляхом складання угод або контрактів, у яких визначаються умови аналізу, типи тестів, обов’язки та терміни.

Основою сортового контролю є фактична фаза тестування, яка може включати морфологічну оцінку, візуальне порівняння та — у більш просунутих лабораторіях — молекулярну діагностику, таку як аналіз ДНК. Ці методи використовуються для підтвердження відповідності насіння за генетичними та фенотиповими ознаками заявленому сорту. Після завершення тестування результати ретельно документуються, аналізуються та інтерпретуються спеціалістами. На основі цього аналізу видається звіт або

сертифікат, який служить офіційним підтвердженням відповідності або невідповідності сорту.

Традиційно цими процесами керували вручну або за допомогою простих цифрових інструментів, таких як електронні таблиці та текстові документи. Однак зростаюча складність і обсяги сортовипробувань, а також зростаючий попит на прозорість і відстежуваність, виявляють обмеження застарілих систем. Помилки в документації, втрата даних, затримки у звітності та порушення зв’язку є поширеними проблемами, які негативно впливають як на ефективність лабораторії, так і на довіру клієнтів.

Для вирішення цих завдань впровадження спеціалізованої інформаційної системи дає значні переваги. Така система централізувала б і автоматизувала весь робочий процес — від реєстрації зразків і створення контракту до керування результатами тестування та створення звітів — зберігаючи при цьому відповідність нормативним стандартам. Впроваджуючи цифрові рішення, адаптовані до конкретних потреб лабораторій сортового контролю, сектор може отримати переваги від підвищеної точності, швидкості роботи та покращеної координації між зацікавленими сторонами, залученими до забезпечення якості насіння [3].

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
SeedSample	sampleID	Унікальний ідентифікатор насіннєвого зразка.
	varietyName	Назва сорту насіння.
	lotNumber	Номер партії насіння.
	origin	Походження насіннєвого зразка (наприклад, країна, регіон).

Таблиця 1.1 (Продовження)

	receptionDate	Дата отримання насіннєвого зразка в лабораторії.
	status	Статус обробки (наприклад, "на перевірці", "перевірено").
Contract	contractID	Унікальний ідентифікатор контракту на проведення тестування.
	seedSampleID	Ідентифікатор насіннєвого зразка, до якого відноситься контракт.
	clientName	Назва компанії або ім'я клієнта, що надіслав зразок.
	termsOfAgreement	Умови угоди щодо тестування (наприклад, терміни, обсяг робіт).
	contractDate	Дата укладення контракту.
TestResult	resultID	Унікальний ідентифікатор результату тестування.
	seedSampleID	Ідентифікатор насіннєвого зразка, до якого відноситься результат.
	testingDate	Дата проведення тестування.
	result	Результат тестування (наприклад, "відповідає", "не відповідає").

1.2 Огляд існуючих рішень

Потреба в ефективному сортовому контролі в лабораторіях випробування насіння призвела до розробки різноманітних програмних рішень, призначених для оптимізації операцій, покращення управління даними та забезпечення відповідності нормативним стандартам. У цьому розділі представлено огляд існуючих систем та інструментів, які зараз використовуються в сільськогосподарському та лабораторному секторах.

SeedManager — це комплексна програмна платформа, спеціально розроблена для лабораторій з тестування насіння, спрямована на підвищення ефективності та точності процесів контролю сортів. Цей веб-додаток оптимізує різні лабораторні операції, дозволяючи користувачам керувати зразками насіння, контрактами та результатами тестів у єдиному середовищі [1].

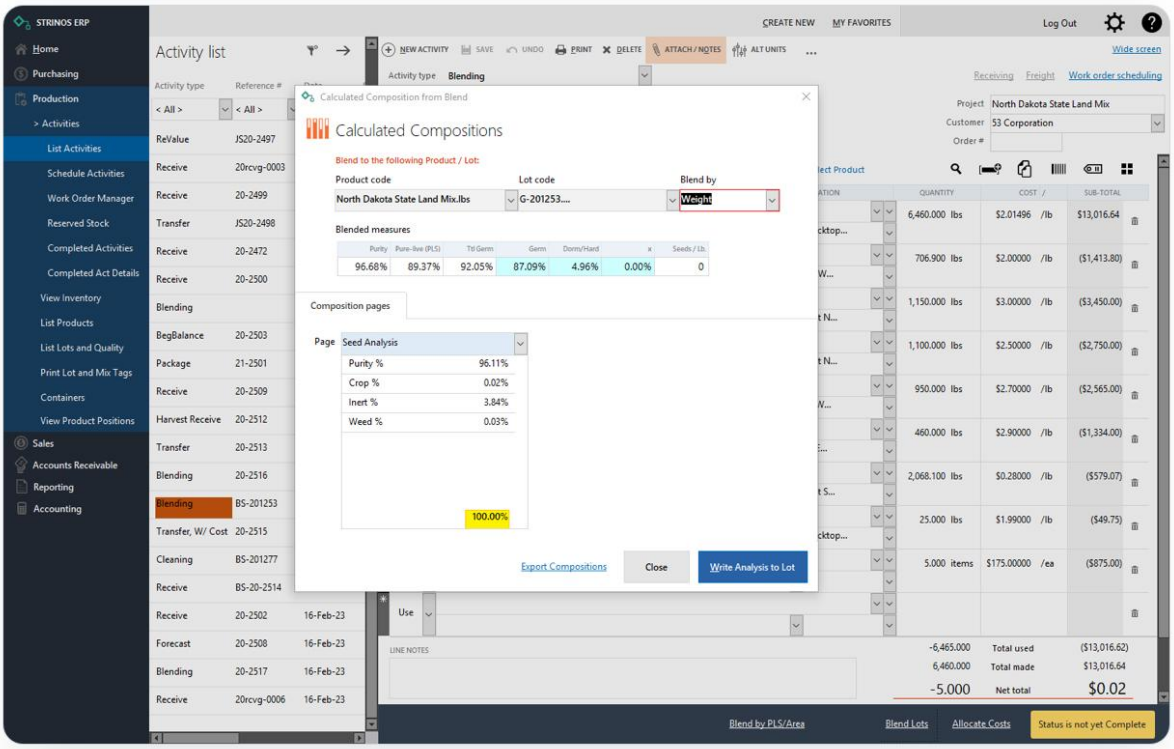

Рис. 1.1 – Інформаційна система “SeedManager”

Однією з ключових особливостей SeedManager є його здатність відстежувати зразки насіння протягом життєвого циклу тестування. Користувачі можуть легко реєструвати вхідні зразки, призначати унікальні ідентифікатори та контролювати їхній статус під час проходження різних етапів тестування. Це відстеження в режимі реального часу гарантує, що лабораторії можуть підтримувати високий рівень організації та нагляду, зменшуючи ймовірність помилок і неправильного управління.

Окрім керування зразками, SeedManager пропонує можливості автоматичного створення звітів. Ця функція дозволяє лабораторіям ефективно

створювати комплексні звіти про випробування, які можна налаштувати відповідно до конкретних нормативних вимог. Завдяки автоматизації процесу звітності SeedManager економить дорогоцінний час персоналу лабораторії, дозволяючи їм зосередитися на критичному аналізі, а не на адміністративних завданнях.

Платформа також містить функції аналітики даних, що надає лабораторіям уявлення про їх операційну продуктивність. Аналізуючи тенденції та показники, пов’язані з тестуванням насіння, лабораторії можуть визначити сфери для вдосконалення, оптимізувати робочі процеси та підвищити загальну продуктивність. Цей підхід, що керується даними, сприяє розвитку культури постійного вдосконалення в лабораторії [1].

Хоча SeedManager пропонує потужний набір функцій, важливо зазначити, що його складність може вимагати від користувачів певного рівня підготовки. Новому персоналу може знадобитися вказівка для ефективної навігації системою, що може стати проблемою для деяких лабораторій. Тим не менш, переваги спрощених операцій і покращеного керування даними часто переважують ці початкові перешкоди.

Підводячи підсумок, SeedManager є ефективним рішенням для лабораторій з тестування насіння, які прагнуть покращити процеси контролю сортів. Завдяки інтеграції відстеження зразків, автоматизованого звітування та аналітики даних в єдину платформу SeedManager дозволяє лабораторіям працювати ефективніше та забезпечує відповідність галузевим стандартам. Таким чином, він є цінним інструментом для лабораторій, які прагнуть підтримувати цілісність і якість тестування насіння.

Розглянемо інше програмне рішення на ринку.

LabWare LIMS — це потужна система управління лабораторною інформацією, призначена для оптимізації лабораторних операцій у різних галузях промисловості, включаючи сільське господарство, фармацевтику та екологічні випробування. Це комплексне рішення забезпечує надійну структуру для керування зразками, даними та робочими процесами,

забезпечуючи відповідність нормативним вимогам і підвищуючи загальну ефективність [2].

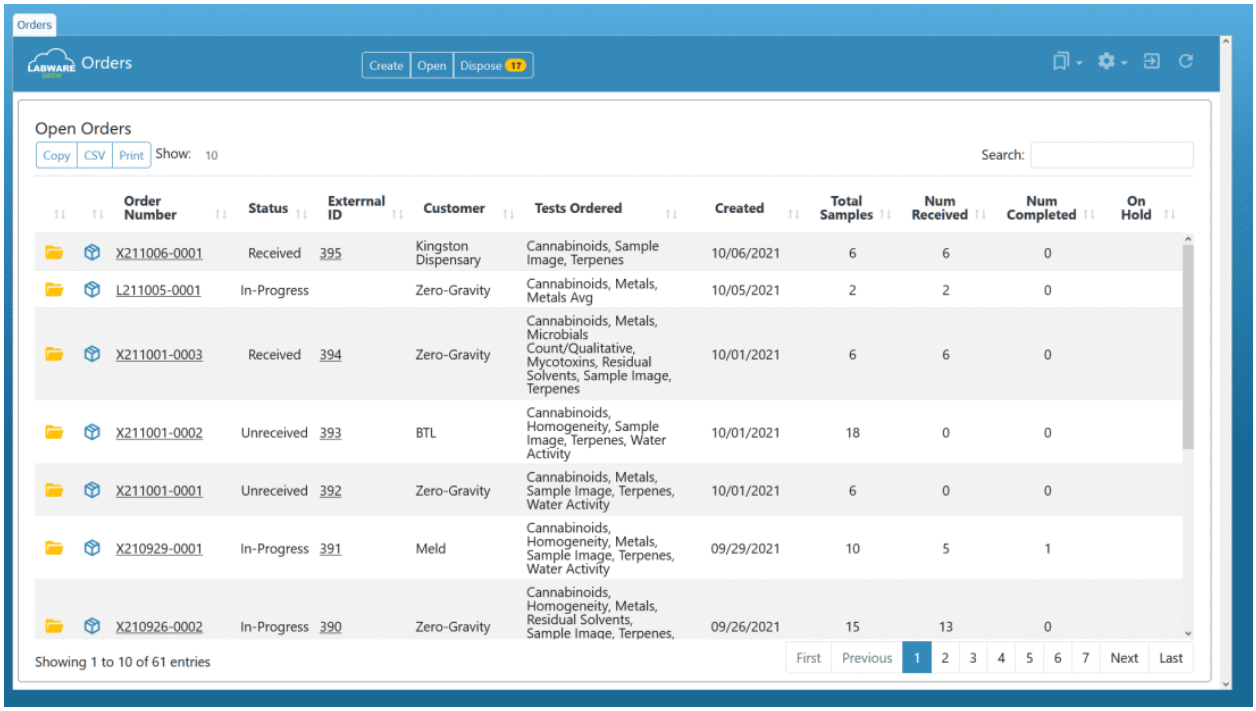

Рис. 1.2 – Інформаційна система “LabWare LIMS”

Однією з видатних особливостей LabWare LIMS є його здатність підтримувати широкий спектр лабораторних процесів, від прийому зразків і відстеження до аналізу даних і звітності. Система дозволяє користувачам керувати складними робочими процесами шляхом автоматизації рутинних завдань, що мінімізує ймовірність людських помилок і підвищує продуктивність. Користувачі можуть легко реєструвати вхідні зразки, призначати унікальні ідентифікатори та відстежувати їхній статус протягом життєвого циклу тестування, гарантуючи, що вся відповідна інформація є легкодоступною [2].

LabWare LIMS також пропонує розширені можливості керування даними, що дозволяє лабораторіям вести детальні записи результатів тестів та іншої важливої інформації. Система підтримує інтеграцію з різними лабораторними приладами, що забезпечує безперебійну передачу даних і зменшує потребу в ручному введенні. Ця інтеграція не тільки економить час,

але й підвищує точність даних, життєво важливий аспект підтримки відповідності галузевим стандартам.

На додаток до своїх основних функціональних можливостей LabWare LIMS надає настроювані інструменти звітування, які дозволяють користувачам створювати комплексні звіти, адаптовані до їхніх конкретних потреб. Ці звіти можуть містити візуалізації, статистичні аналізи та документацію щодо відповідності, що полегшує лабораторіям передачу результатів зацікавленим сторонам і регуляторним органам.

Хоча LabWare LIMS пропонує широкі можливості та гнучкість, важливо зазначити, що система може потребувати значних інвестицій з точки зору як часу, так і ресурсів для впровадження. Лабораторії повинні бути готові інвестувати в навчання персоналу та гарантувати, що їх IT-інфраструктура може підтримувати можливості системи. Незважаючи на ці проблеми, переваги підвищеної операційної ефективності, покращеної цілісності даних і спрощених процесів відповідності роблять LabWare LIMS цінним активом для лабораторій, які прагнуть до досконалості.

Таким чином, LabWare LIMS є провідним рішенням для лабораторій, які прагнуть оптимізувати свою роботу та забезпечити дотримання нормативних стандартів. Його комплексні функції в поєднанні з можливістю налаштовувати робочі процеси та звітність дають змогу лабораторіям працювати ефективніше та отримувати високоякісні результати [2].

1.3 Постановка завдання

Керівник лабораторії несе відповідальність за підтримку всебічного огляду ключових показників, пов’язаних із сортовим контролем і взаємодією з клієнтами, таких як кількість оброблених зразків насіння, точність результатів випробувань і час для звітності. Для ефективного управління цією інформацією запропонована інформаційна система створить надійну базу

даних, яка фіксує всі важливі дані, необхідні для створення персоналізованих сповіщень і інформації щодо ефективності процесу сортовипробування.

Основна програмна система, що розробляється, дозволить персоналу лабораторії швидко та ефективно отримувати доступ до критичної інформації, зокрема:

- ефективність методів і протоколів сортовипробування;
- точність дотримання стандартів сертифікації насіння;
- кількість зразків насіння, перевірених за визначені періоди часу.

Програма працюватиме через інтуїтивно зрозумілий графічний інтерфейс користувача, який дозволяє користувачам легко переміщатися між різними параметрами. Це налаштування дозволить персоналу лабораторії легко вводити та змінювати наявні записи зразків, забезпечуючи швидке та точне внесення будь-яких змін або оновлень.

Розширюючи можливості інформаційної системи, лабораторія підвищить ефективність роботи, підвищить точність сортоконтролю та пришвидшить роботу клієнтів. Кінцева мета полягає в тому, щоб підвищити якість послуг з тестування насіння, тим самим сприяючи більшій довірі між зацікавленими сторонами та сприяючи загальному підвищенню продуктивності сільського господарства.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. система сприятиме ефективному відстеженню та управлінню зразками насіння протягом усього процесу тестування, від отримання до звітності. Це гарантує, що кожен зразок точно реєструється та контролюється, зменшуючи ризик помилок і неправильного розташування;

2. завдяки автоматизації створення звітів про випробування система заощадить персонал лабораторії значний час і зусилля. Ця функція дозволить швидше поширювати результати клієнтам і регуляторним органам, покращуючи загальну комунікацію;
3. користувачі отримають миттєвий доступ до важливої інформації щодо стану зразка, результатів тестування та історичних даних. Ця функція підтримує прийняття обґрунтованих рішень і дозволяє персоналу лабораторії оперативно відповідати на запити клієнтів;
4. система включатиме функції, які забезпечують дотримання галузевих стандартів і нормативних вимог. Підтримуючи точні записи та документацію, лабораторії можуть легко полегшити аудити та інспекції;
5. додаток надаватиме аналітичні інструменти, які дозволять лабораторіям оцінювати свою продуктивність з часом, визначати тенденції та оптимізувати процеси тестування. Такий підхід, який базується на даних, може призвести до підвищення ефективності та результативності сортового контролю;
6. інтуїтивно зрозумілий дизайн системи дозволить користувачам легко переміщатися між різними функціями, скорочуючи криву навчання для нових співробітників і підвищуючи загальну задоволеність користувачів.

Нефункціональні вимоги:

1. інформаційна система буде розроблена з урахуванням зростання лабораторії, дозволяючи додавати нових користувачів, зразки та методи тестування без зниження продуктивності. Ця масштабованість гарантує адаптацію системи до змінних потреб лабораторії;

2. впровадження надійних заходів безпеки захистить конфіденційні дані, пов’язані зі зразками насіння та результатами тестів. Автентифікація користувачів і засоби контролю доступу забезпечать доступ до конкретної інформації тільки уповноваженому персоналу, таким чином зберігаючи цілісність і конфіденційність даних;
3. система буде розроблена з упором на надійність, що гарантуватиме її безперебійну роботу без неочікуваних простоїв. Ця надійність має вирішальне значення для підтримки роботи лабораторії та задоволення очікувань клієнтів [4];
4. оптимізація продуктивності системи забезпечить швидкий час відгуку та ефективну обробку даних, що дозволить персоналу лабораторії ефективно працювати без затримок;
5. інформаційна система буде розроблена з урахуванням зручності обслуговування, що дозволить легко оновлювати та модифікувати за потреби. Це гарантує, що система буде відповідати технологічним досягненням і нормативним змінам.

Завдяки підвищенню ефективності роботи та скороченню ручних процесів система може призвести до економії коштів для лабораторії. Очікується, що інвестиції в інформаційну систему принесуть довгострокові вигоди завдяки підвищенню продуктивності та зниженню рівня помилок.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача Інформаційної системи для проведення лабораторного сортового контролю повинен надавати перевагу зручному дизайну, який забезпечує інтуїтивну навігацію, дозволяючи користувачам ефективно виконувати завдання, не вимагаючи тривалого навчання. Чіткий макет із добре організованими меню та параметрами покращить зручність використання та полегшить робочий процес.

В основі інтерфейсу має бути комплексна інформаційна панель, яка надає користувачам швидкий знімок ключових показників, включаючи кількість зразків, які зараз обробляються, статуси їх тестування та останні дії в лабораторії. Ця функція дозволить користувачам миттєво контролювати продуктивність лабораторії, гарантуючи, що вони залишатимуться поінформованими про поточні операції.

Для адаптації різних пристроїв користувальницький інтерфейс має бути адаптивним і безперебійно працювати на настільних комп'ютерах, планшетах і смартфонах. Ця гнучкість має вирішальне значення, оскільки дозволяє користувачам отримувати доступ до системи з різних місць і пристроїв, підвищуючи загальну доступність програми.

Спеціальний модуль керування зразками необхідний для ефективної обробки зразків насіння. Цей розділ має дозволити користувачам легко додавати, редагувати та відстежувати зразки, маючи потужні параметри пошуку та фільтрування, які дозволяють швидко ідентифікувати конкретні зразки на основі різних критеріїв, таких як ідентифікатор зразка, назва сорту та поточний статус.

Окрім керування вибірками, інтерфейс має сприяти автоматизованому створенню звітів. Користувачам потрібні інструменти, які дозволять їм вибирати параметри звіту, налаштовувати формати та експортувати результати в файли різних типів, наприклад PDF і Excel. Цей функціонал значно скоротить час, витрачений на адміністративні завдання.

Форми введення даних в інтерфейсі мають бути розроблені для простоти використання, що дозволяє зрозуміло вводити й оновлювати інформацію, пов'язану зі зразками, результатами випробувань і контрактами. Включення валідаційних перевірок допоможе забезпечити точність і повноту даних, зменшуючи ймовірність помилок.

Ефективна система сповіщень і сповіщень життєво необхідна для інформування користувачів про важливі події. Ця функція повинна сповіщати користувачів про найближчі терміни тестування зразків, вказувати, коли

результати будуть готові для перегляду, і попереджати їх про будь-які проблеми, які потребують негайної уваги.

Для ефективної підтримки користувачів спеціальний розділ довідки має бути легко доступним в інтерфейсі. Цей ресурс має надавати вказівки щодо системних функцій, поради щодо усунення несправностей і контактну інформацію для технічної підтримки, якщо це необхідно.

Управління ролями користувачів — ще один важливий аспект інтерфейсу, що забезпечує безпечну автентифікацію користувачів і контроль доступу на основі ролей. Ця функція гарантує, що різні користувачі, такі як лаборанти, менеджери та адміністратори, можуть отримати доступ до певних функцій, адаптованих до їхніх ролей, таким чином зберігаючи безпеку та цілісність даних.

Візуальне представлення даних також є важливим, оскільки система повинна містити діаграми та графіки, які допомагають користувачам ефективно візуалізувати тенденції та показники ефективності. Ця можливість покращить аналіз даних і допоможе користувачам приймати зважені рішення.

Інтерфейс має бути оптимізовано для продуктивності, забезпечуючи швидке завантаження та оперативну взаємодію. Зводячи до мінімуму затримки, система дозволить користувачам виконувати завдання ефективно, сприяючи позитивному загальному досвіду.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) — це стандартизована мова моделювання, призначена для візуалізації, специфікації, конструювання та документування різних елементів програмних систем. Це важливий інструмент для подолання розриву між системними вимогами та дизайном, забезпечуючи чітку комунікацію між розробниками, архітекторами та зацікавленими сторонами щодо структури та поведінки системи [5].

UML надає різноманітні графічні нотації та типи діаграм, які відображають різні аспекти програмної системи. Серед них структурні діаграми зосереджені на статичних компонентах і організації системи. Діаграми класів є особливо важливими, оскільки вони зображують різні класи в системі, описуючи їхні атрибути, методи та взаємозв’язки. Діаграми компонентів ілюструють організацію та залежності між компонентами програмного забезпечення, тоді як діаграми розгортання показують, як артефакти розподіляються між фізичними вузлами, деталізуючи задіяні конфігурації апаратного та програмного забезпечення.

З іншого боку, поведінкові діаграми розглядають динамічні аспекти системи, підкреслюючи, як вона поводиться та взаємодіє з часом. Діаграми варіантів використання важливі для фіксації взаємодії між користувачами або акторами та системою, окреслюючи її функціональні можливості з точки зору кінцевих користувачів. Діаграми послідовності додатково деталізують ці взаємодії, зображуючи, як об’єкти спілкуються в конкретних сценаріях, підкреслюючи послідовність повідомлень, якими вони обмінюються. Діаграми діяльності забезпечують ширше уявлення про робочі процеси в системі, ілюструючи послідовність дій і моментів прийняття рішень, залучених до різних процесів.

Діаграми взаємодії, такі як діаграми співпраці, доповнюють ці представлення, зосереджуючись на потоці повідомлень між об’єктами, демонструючи ролі різних учасників у взаємодії [5].

Широке застосування UML у розробці програмного забезпечення можна пояснити його здатністю забезпечувати чітке та візуальне представлення складних систем. Цей візуальний підхід полегшує зацікавленим сторонам усвідомлення вимог і дизайнерських рішень, що зрештою сприяє кращому розумінню та співпраці між членами команди. Використовуючи UML, групи розробників можуть забезпечити узгодженість документації та підвищити загальну якість створюваного програмного забезпечення.

Однією з ключових сильних сторін UML є його адаптивність, оскільки його можна застосовувати в різних областях і методологіях розробки, включаючи гнучкі практики, об’єктно-орієнтоване проєктування та системну інженерію. Його універсальність позиціонує UML як цінний ресурс для розробників, аналітиків і керівників проєктів протягом життєвого циклу розробки програмного забезпечення.

Підсумовуючи, UML є ключовим елементом сучасної інженерії програмного забезпечення, що пропонує спільну мову для моделювання та візуалізації систем. Сприяючи ефективній комунікації та розумінню між усіма зацікавленими сторонами, залученими до процесу розробки, UML значно сприяє успіху програмних проєктів.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграми варіантів використання є життєво важливим компонентом уніфікованої мови моделювання (UML), який використовується для ілюстрації функціональних вимог системи. Вони забезпечують візуальне представлення того, як різні користувачі, відомі як актори, взаємодіють із системою для досягнення конкретних цілей або

завдань. Цей тип діаграм особливо корисний на ранніх стадіях розробки програмного забезпечення, оскільки він допомагає охопити та уточнити вимоги з точки зору кінцевих користувачів [6].

В основі діаграми варіантів використання знаходяться актори та варіанти використання. Актори представляють зовнішні сутності, які взаємодіють із системою, до яких можуть входити користувачі, інші системи або пристрої. Кожен актор зображений у вигляді фігурки або прямокутника з позначками, що вказує на їх роль у взаємодії.

Варіанти використання, представлені овалами, інкапсулюють певні функціональні можливості або процеси в системі, з якими учасники можуть взаємодіяти. Кожен варіант використання описує окрему мету або результат, якого прагне досягти учасник, наприклад «Надіслати зразок», «Створити звіт» або «Переглянути результати тесту». Відносини між акторами та варіантами використання зображені лініями, що демонструє, як актори ініціюють або беруть участь у різних функціях системи.

Окрім базових зв’язків, діаграми варіантів використання також можуть ілюструвати складніші взаємодії за допомогою зв’язків, таких як включення та розширення. Відношення включення означає, що один варіант використання є необхідною частиною іншого, тоді як зв’язок розширення вказує на необов’язкову поведінку, яка може покращити основний варіант використання.

Основною перевагою діаграм варіантів використання є їх здатність чітко та зрозуміло передавати функціональність системи. Вони надають зацікавленим сторонам загальний огляд можливостей системи, полегшуючи визначення вимог і гарантуючи, що всі необхідні функції охоплені. Це візуальне представлення сприяє співпраці між членами команди, допомагаючи узгодити процес розробки з потребами та очікуваннями користувачів.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

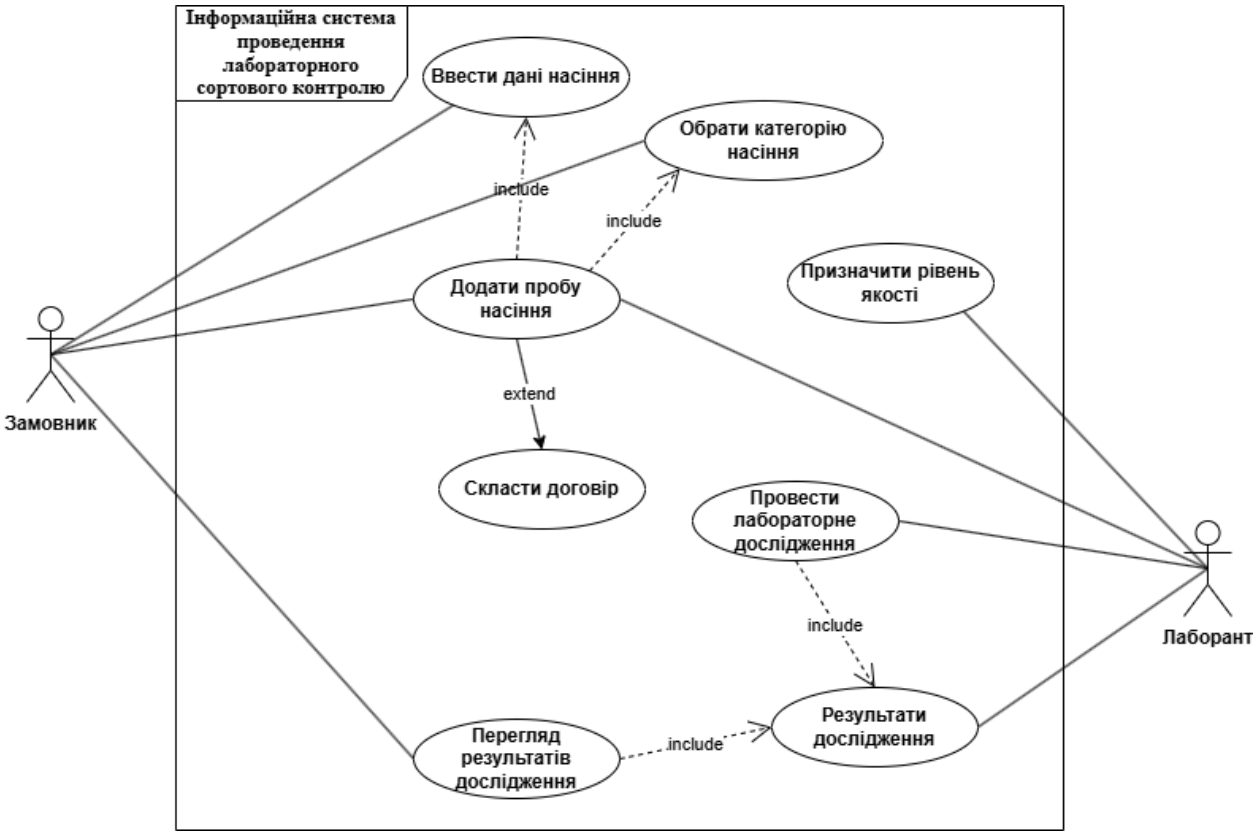

Рис. 2.1 – Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Замовник”;
- “Лаборант”.

Актор «Замовник» включає такі прецеденти:

- ввести дані насіння;
- обрати категорію насіння;
- додати пробу насіння;
- скласти договір;
- перегляд результатів дослідження.

Актор «Лаборант» включає такі прецеденти:

- провести лабораторне дослідження;
- результати дослідження;
- призначити рівень якості;
- додати пробу насіння.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Випадок використання: додати пробу насіння

Актор: Замовник

Опис: цей варіант використання описує процес, за допомогою якого клієнт подає новий зразок насіння для сортового тестування в лабораторії. Замовником зазвичай є виробник насіння, сільськогосподарський працівник або будь-яка інша організація, яка прагне перевірити якість і відповідність своїх зразків насіння.

Основний потік:

1. клієнт переходить до розділу «Додати зразок» програми;
2. система пропонує клієнту форму для заповнення. Клієнт надає важливі деталі;
3. за потреби клієнт завантажує будь-які необхідні документи, такі як сертифікати походження або результати попередніх тестів;
4. Клієнт перевіряє введену інформацію на предмет точності та повноти;
5. клієнт натискає кнопку «Надіслати», щоб завершити додавання зразка;
6. система обробляє запит і призначає унікальний ідентифікатор зразка. Клієнту відображається повідомлення про підтвердження, яке вказує на те, що зразок успішно додано;
7. система надсилає сповіщення на зареєстровану адресу електронної пошти клієнта, підтверджуючи подання та надаючи такі деталі, як ідентифікатор зразка та очікуваний графік тестування.

Альтернативні потоки:

1. якщо будь-які обов’язкові поля відсутні або містять недійсні дані, система запропонує клієнту виправити помилки перед повторним надсиланням форми;

2. якщо завантажені документи не відповідають заданим критеріям (наприклад, розмір файлу, формат), система сповіщає клієнта про необхідність повторного завантаження документів.

Цей приклад використання ілюструє простий процес для клієнтів, щоб додати зразок рослини до системи лабораторії. Забезпечуючи ефективний і зручний інтерфейс, інформаційна система оптимізує процес подання зразків, покращуючи загальний досвід для клієнта, забезпечуючи при цьому точний збір даних для сортового тестування.

Розглянемо ще один сценарій використання.

Випадок використання: проведення лабораторних досліджень

Актор: лаборант

Опис: цей варіант використання описує процес, за допомогою якого лаборант проводить дослідження та тестування зразків насіння для визначення їх сортових характеристик і якості. Це передбачає аналіз зразків відповідно до встановлених протоколів і документування результатів у системі.

Основний потік:

1. лаборант переходить до розділу «Зразки» системи та отримує список зразків, які очікують на тестування;
2. лаборант вибирає конкретний зразок насіння зі списку для проведення дослідження;
3. система відображає детальну інформацію про вибраний зразок, включаючи його унікальний ідентифікатор, назву сорту та будь-які відповідні примітки чи історію;
4. лаборант проводить необхідні випробування зразка насіння;
5. після завершення тестів лаборант вводить результати в систему;
6. після завершення всіх тестів і запису результатів помічник натискає кнопку «Завершити дослідження», щоб надіслати результати;

7. система оновлює статус зразка насіння на «Тест завершено» та створює звіт, у якому підсумовуються проведені дослідження та отримані результати;
8. система автоматично сповіщає відповідних зацікавлених сторін (наприклад, клієнта або менеджера), що лабораторне дослідження завершено та що результати доступні для перегляду.

Альтернативні потоки:

1. якщо під час тестування виникають будь-які проблеми (наприклад, несправність обладнання або недостатній зразок), асистент може записати звіт про помилку в системі та повідомити керівника для подальших дій;
2. якщо лаборант намагається завершити дослідження, не ввівши всі необхідні результати, система запропонує йому доповнити відсутню інформацію перед подачею.

Цей приклад використання ілюструє процес, за допомогою якого лаборант проводить дослідження зразків насіння в інформаційній системі. Спрощуючи організоване введення даних і надаючи легкий доступ до інформації про зразки, система підвищує ефективність і точність лабораторних випробувань, зрештою підтримуючи цілі сортового контролю та гарантії якості.

2.2.2 Діаграма послідовності. Діаграми послідовностей — це тип діаграм взаємодії в уніфікованій мові моделювання (UML), які ілюструють, як об’єкти взаємодіють у певній послідовності протягом певного часу. Вони особливо корисні для візуалізації потоку повідомлень між різними компонентами системи, фіксуючи динамічну поведінку системи, коли вона виконує певний варіант використання [7].

Основна мета діаграми послідовності — відобразити порядок взаємодії між різними об’єктами, показуючи, як вони співпрацюють для досягнення певної мети. Ця візуалізація допомагає зрозуміти конкретну

поведінку системи під час певного сценарію, полегшуючи розробникам і зацікавленим сторонам зрозуміти основну логіку та потік операцій.

Діаграми послідовності важливі для розробки програмного забезпечення з кількох причин. Вони забезпечують чіткий і організований спосіб документування взаємодій, необхідних для виконання сценарію використання, полегшуючи командам виявлення потенційних проблем і оптимізацію дизайну системи. Візуалізуючи порядок операцій, розробники можуть краще зрозуміти потік даних і управління, що призводить до покращення стратегій впровадження та тестування.

Крім того, діаграми послідовності служать цінним інструментом спілкування між зацікавленими сторонами, оскільки вони передають складні взаємодії в зручному форматі. Їх можна використовувати протягом життєвого циклу розробки програмного забезпечення, від початкового етапу проєктування до впровадження та тестування, гарантуючи, що поведінка системи відповідає вимогам і очікуванням користувачів.

Таким чином, діаграми послідовності є важливим інструментом в UML, який фіксує динамічну взаємодію між об’єктами в системі. Надаючи візуальне представлення того, як ці взаємодії розгортаються з часом, діаграми послідовності покращують розуміння, спілкування та документування поведінки системи [7].

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- замовник;
- лаборант;
- проби насіння;
- договір;
- результати дослідження.

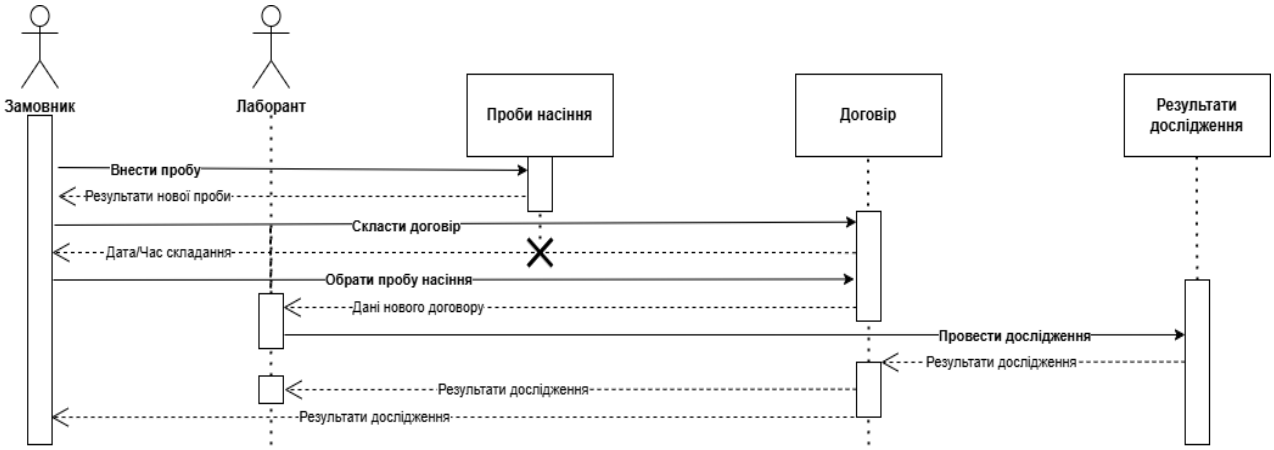

Рис. 2.2 – Діаграма послідовності

Об'єкт «Замовник» ініціює запит до об'єкта «Проби насіння», вносячи нову пробу вказавши всі необхідні параметри. У відповідь він отримує результати нової проби. Після цього він створює договір вказавши потрібні йому проби. Ця інформація передається до об'єкта «Результати дослідження», де необхідні дані заповнюються для подальшої обробки.

На наступному етапі об'єкт «Лаборант» формує запит до об'єкта «Результати дослідження» на проведення дослідження. У відповідь система обробляє запит і надсилає результат дослідження.

2.2.3 Діаграма активності. Діаграми діяльності — це тип діаграми поведінки в уніфікованій мові моделювання (UML), яка представляє потік дій у системі. Вони особливо корисні для моделювання динамічних аспектів процесу, ілюструючи, як виконуються завдання та послідовність, у якій вони відбуваються. Діаграми діяльності допомагають візуалізувати робочі процеси, що робить їх ефективним інструментом для розуміння та вдосконалення процесів.

Основна мета діаграми діяльності — відобразити послідовність дій і рішень, які мають місце в певному сценарії або випадку використання. Вони надають чітке уявлення про потік контролю від однієї діяльності до

іншої, висвітлюючи шляхи, якими можна скористатися на основі конкретних умов [8].

Діаграми діяльності важливі для розробки програмного забезпечення з кількох причин. Вони забезпечують огляд процесів на високому рівні, полегшуючи зацікавленим сторонам розуміння складних робочих процесів і визначення областей для вдосконалення. Візуалізуючи потік дій, розробники можуть виявляти вузькі місця, зайві завдання та неефективність, що призводить до кращої оптимізації процесів.

Крім того, діаграми діяльності служать цінними інструментами документування, допомагаючи командам чітко спілкуватися між зацікавленими сторонами щодо робочих процесів. Їх можна використовувати протягом життєвого циклу розробки програмного забезпечення, від збору вимог і проєктування до впровадження та тестування, гарантуючи, що всі члени команди мають спільне розуміння процесів, що моделюються.

Таким чином, діаграми активності є важливими інструментами в UML для моделювання та візуалізації потоку діяльності в системі. Забезпечуючи чітке представлення робочих процесів, вони покращують розуміння, комунікацію та документування процесів, що зрештою сприяє більш ефективній розробці програмного забезпечення.

Розроблена діаграма активності представлена на рис. 2.3.

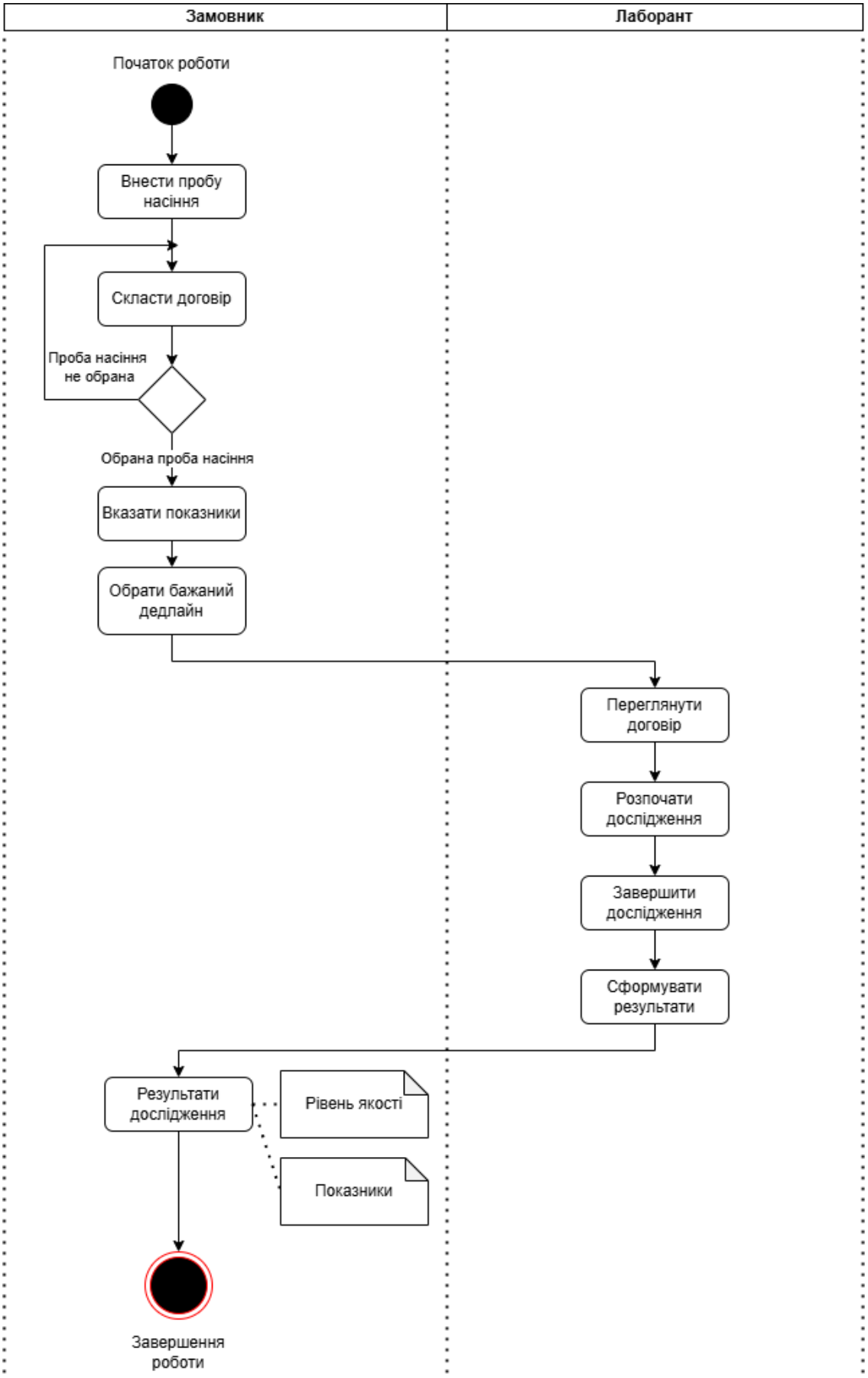

Рис. 2.3 Діаграма активності

2.3 Абстракції предметної області

Під час розробки програмного забезпечення важливим етапом є формування абстракцій предметної області. Це поняття означає узагальнене подання реальних об'єктів, процесів та явищ, які є частиною конкретної сфери діяльності, що автоматизується. Абстракція дозволяє розробникам зосередитись на ключових характеристиках об'єктів, відкидаючи другорядні деталі, які не мають прямого відношення до роботи системи. Саме завдяки такому підходу можна ефективно аналізувати структуру предметної області, встановлювати логічні зв’язки між об’єктами, будувати моделі даних та описувати функціональні сценарії взаємодії користувача з системою.

У випадку інформаційної системи для лабораторного сортового контролю, предметна область охоплює процеси збирання зразків насіння, оформлення контрактів між замовниками та лабораторією, проведення досліджень, а також фіксацію результатів аналізу. В цій області важливу роль відіграють такі сутності як «Зразок», «Контракт», «Замовник», «Лабораторний асистент», «Результати досліджень». Кожна з цих сутностей може бути описана у вигляді абстракції з чітко визначеним набором атрибутів, які відповідають основним характеристикам об'єкта. Наприклад, зразок насіння характеризується сортом, датою відбору, походженням; контракт – номером, датою підписання, переліком зразків; замовник – ім’ям, контактною інформацією, установою, яку він представляє.

Використання абстракцій дозволяє створити логічну модель системи, яка буде зрозумілою як для технічних спеціалістів, так і для користувачів, що працюють у відповідній галузі. Це особливо важливо на етапах моделювання UML-діаграм, проєктування бази даних та реалізації інтерфейсу користувача. Завдяки абстракціям розробник може легко визначити, які об’єкти мають бути реалізовані у вигляді класів, які між ними існують зв’язки, які атрибути необхідно зберігати, а також які дії або сценарії передбачені для кожного об’єкта [9].

Крім того, правильно побудовані абстракції сприяють масштабованості системи. У разі зміни вимог або розширення функціоналу, розробник зможе безболісно внести корективи до логіки системи, не порушуючи її цілісності. Таким чином, абстракції виступають своєрідним каркасом, на основі якого формується програмна архітектура (рис. 2.4).

Рис. 2.4 – Абстракції предметної області

Узагальнюючи, можна сказати, що абстракції предметної області — це інструмент, який забезпечує перехід від реального світу до програмної моделі. Вони допомагають структурувати знання про досліджувану область, оптимізувати процес розробки та гарантувати, що створене програмне забезпечення буде повністю відповідати поставленим завданням і потребам кінцевих користувачів.

2.4 Діаграма класів

Діаграма класів — це фундаментальний компонент об’єктно-орієнтованого моделювання, який візуалізує структуру системи шляхом відображення її класів, атрибутів, методів і зв’язків між ними. У контексті інформаційної системи, розробленої для лабораторного сортового контролю,

діаграма класів відіграє ключову роль у окресленні того, як різні компоненти системи взаємодіють на концептуальному рівні.

Діаграма представляє основні сутності, такі як Зразок, Контракт, Замовник, Лаборант, Результат дослідження та Тест. Кожен із цих класів інкапсулює як дані, так і поведінку, що мають відношення до аналогів у реальному світі. Наприклад, клас Sample може містити такі атрибути, як ідентифікатор зразка, сорт рослини, дата збору та статус, а також мати методи для оновлення або перевірки інформації. Клас контракту зазвичай зберігає номер контракту, пов’язаного клієнта, список зразків і дату підписання. Подібним чином клас ResearchResult містить дані, пов’язані з лабораторним тестуванням, такі як дата тесту, значення результатів, інтерпретація та посилання на відповідний зразок

Відносини між класами також візуалізуються на схемі. Наприклад, існує зв’язок між клієнтом і контрактом, коли один клієнт може мати кілька контрактів. Контракт може бути пов’язаний з кількома Зразками, і кожен Зразок може відповідати одному або декільком Результатам дослідження. Залежно від складності домену та бажаної архітектури також можна застосовувати успадкування, композицію та агрегацію [10].

Діаграма класів служить не тільки інструментом планування, але й документацією для розробників, допомагаючи підтримувати ясність під час впровадження та майбутніх оновлень системи. Чітко визначаючи структуру та взаємодію ключових компонентів, ця модель підтримує надійну, масштабовану розробку програмного забезпечення, яке точно відображає бізнес-процеси лабораторного сортовипробування.

Для цього веб-сайту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 2.5).

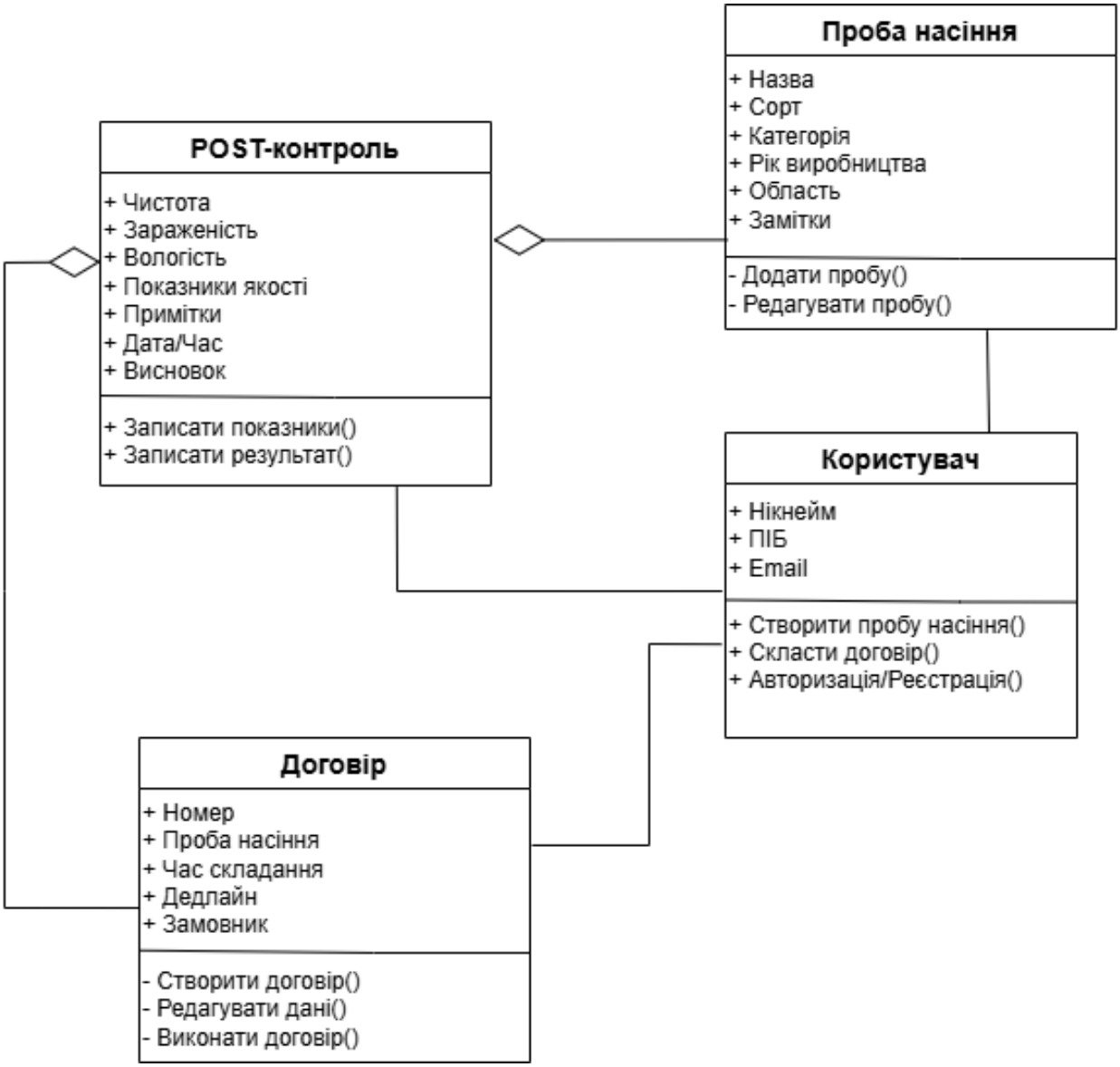

Рис. 2.5 – Діаграма класів

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних надає концептуальний план структури даних інформаційної системи, ілюструючи, як організовані елементи даних і як вони пов’язані один з одним, не заглиблюючись у технічні особливості технологій зберігання чи бази даних. Цей тип моделі діє як перехідний рівень між бізнес-вимогами та майбутньою архітектурою бази даних, допомагаючи визначити, яку інформацію потрібно зібрати та як ця інформація взаємопов’язана в системі.

За своєю суттю логічна модель даних зосереджена на ідентифікації ключових об’єктів у предметній області та з’ясуванні їхніх взаємозв’язків. Ці об’єкти представляють реальні об’єкти або концепції, пов’язані з системою, наприклад зразки насіння, клієнти, контракти, лаборанти, тести та результати досліджень. Кожна сутність описується за допомогою набору атрибутів, які відображають конкретні точки даних, які потрібно зберегти. Наприклад, зразок насіння може містити такі деталі, як ідентифікатор, назва сорту та дата збору.

Важливо те, що логічна модель описує, як ці сутності взаємодіють. Наприклад, один клієнт може бути пов’язаний з декількома контрактами, кожен контракт може містити кілька зразків насіння, і кожен зразок може бути підданий ряду лабораторних тестів. Результати цих тестів потім прив’язуються до окремих зразків, з яких вони походять. Ця мережа зв’язків утворює структурну основу моделі даних [11].

Іншим важливим аспектом є нормалізація, яка використовується для забезпечення логічної узгодженості даних, уникнення надмірності та збереження цілісності. На відміну від фізичних моделей даних, логічні моделі залишаються відокремленими від технічних деталей, таких як типи даних,

індексування або специфічні для системи обмеження. Натомість наголос робиться на точному представленні бізнес-логіки та потоку даних у системі.

У контексті інформаційної системи лабораторного сортового контролю логічна модель даних пропонує структурований спосіб концептуалізації зв’язку між клієнтами, контрактами, зразками, процедурами тестування та результатами. Це представлення високого рівня закладає основу для побудови фактичної бази даних — такої, яка реалізована за допомогою MySQL — і направляє розробників у перетворенні бізнес-потреб у функціональне, масштабоване серверне рішення.

Логічна модель системи представлена на рисунку 3.1:

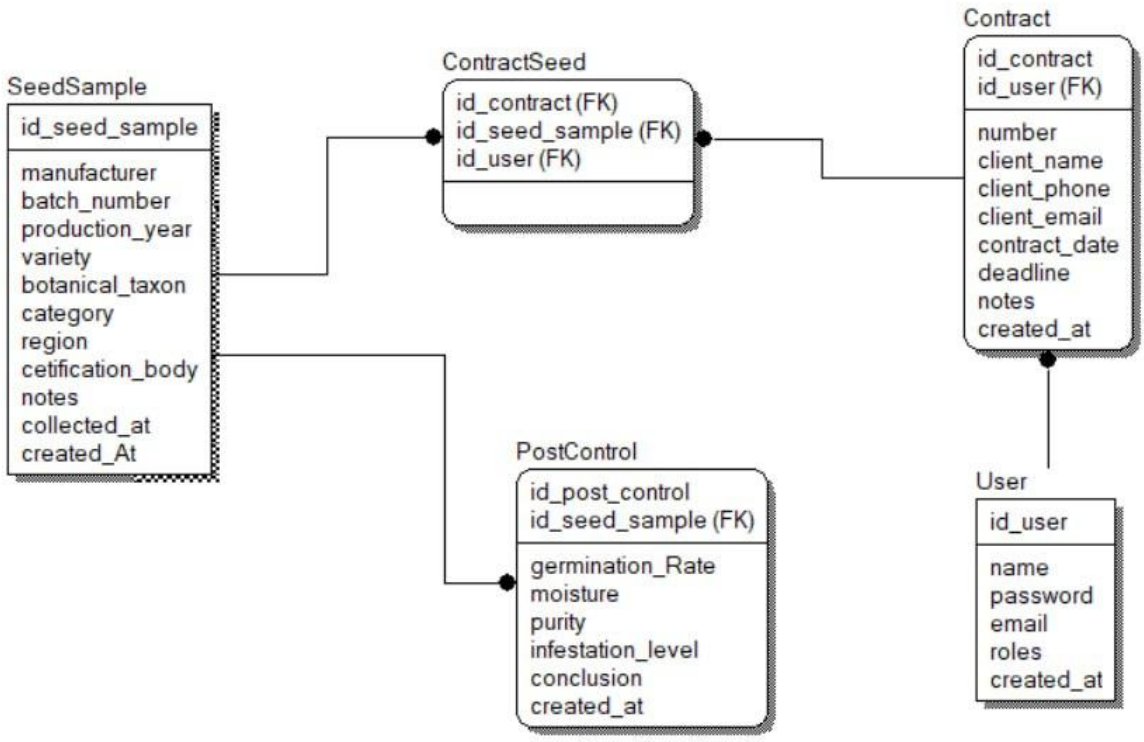

Рис. 3.1 – ER-діаграма

Логічна модель складається з таких сутностей:

- user;
- contract;
- contractSeed;
- seedSample;
- postControl.

Опис структури бази даних

User:

1. id — унікальний ідентифікатор користувача (тип: ціле число);
2. name — ім’я користувача або логін (тип: рядок, до 255 символів);
3. password — хеш паролю користувача (тип: рядок, до 255 символів);
4. email — електронна пошта користувача (тип: рядок, до 255 символів, може бути null);
5. roles — масив ролей користувача (тип: масив, може бути null; наприклад: ROLE_ADMIN, ROLE_USER);
6. createdAt — дата та час створення акаунту (тип: незмінна дата/час, DateTimeImmutable).

ContractSample:

1. id — унікальний ідентифікатор зв’язку (тип: ціле число);
2. contract — посилання на договір, до якого належить проба (тип: зв’язок ManyToOne з Contract);
3. seedSample — посилання на конкретну пробу насіння, включену в договір (тип: зв’язок ManyToOne з SeedSample).

PostControl:

1. id — унікальний ідентифікатор записи (тип: ціле число);
2. seedSample — зв'язок "один до одного" з пробою насіння, для якої проводиться POST-контроль;
3. germinationRate — показник схожості, що відображає відсоток пророслого насіння (тип: ціле число, може бути null);
4. moisture — вологість насіння у відсотках (тип: дійсне число, може бути null);
5. purity — чистота партії насіння, тобто вміст основної культури у вибірці (тип: дійсне число, може бути null);

6. infestationLevel — рівень зараженості насіння (наприклад, шкідниками або грибками) (тип: дійсне число, може бути null);
7. conclusion — висновок за результатами POST-контролю (тип: рядок довжиною до 5000 символів; наприклад: «Пройдено», «Не пройдено»);
8. createdAt — дата та час створення запису (тип: незмінна дата/час, може бути null).

SeedSample:

1. id — унікальний ідентифікатор проби;
2. sampleCode — внутрішній або зовнішній ідентифікаційний код проби;
3. collectionDate — дата відбору проби;
4. origin — місце походження насіння (наприклад, господарство або контрагент);
5. postControl — зв’язок з післяконтролем (один до одного).

Contract:

1. id — унікальний ідентифікатор договору;
2. number — номер договору;
3. date — дата укладення договору;
4. clientName — назва замовника або контрагента;
5. seedSamples — список проб насіння, які відносяться до цього договору (зв’язок один до багатьох).

У розробленій інформаційній системі реалізовано логічні зв’язки між основними сутностями. Сутність SeedSample (проба насіння) є центральною та пов’язана із сутністю Contract (договір) через проміжну таблицю ContractSample, що реалізує зв’язок типу "багато-до-багатьох", тобто одна проба може входити до кількох договорів, а один договір може включати кілька проб. Також SeedSample має зв’язок "один-до-багатьох" з сутністю PostControl, що дозволяє зберігати результати контролю якості проб після укладання договорів. Сутність User використовується для аутентифікації

користувачів системи та не має прямих зовнішніх зв’язків із іншими таблицями, однак забезпечує доступ до функціоналу відповідно до ролей.

3.2 Вибір системи управління базою даних та її реалізація

При розробці інформаційної системи лабораторного сортоконтролю одним із найважливіших технічних рішень є вибір відповідної системи керування базами даних (СУБД). СУБД служить основою для зберігання, організації, пошуку та цілісності даних у всій системі, забезпечуючи ефективне керування всіма об’єктами, такими як початкові зразки, клієнти, контракти, лабораторні тести та результати.

Враховуючи вимоги до системи, включаючи масштабованість, надійність, легкість інтеграції з веб-технологіями та підтримку структурованих мов запитів, MySQL було обрано як кращу СУБД. MySQL — це система керування реляційною базою даних із відкритим вихідним кодом, яка пропонує надійну підтримку багатокористувацького доступу, надійну безпеку даних і чудову продуктивність за помірних і високих навантажень даних. Його сумісність із PHP і Symfony (які використовувалися при розробці програми) забезпечує безперебійну інтеграцію серверної частини та спрощує загальний процес впровадження.

Процес впровадження починається зі створення логічної моделі даних, яка потім транслюється у фізичну схему бази даних. Ця схема містить визначення таблиць для основних сутностей, таких як «Клієнти», «Контракти», «SeedSamples», «LabAssistants», «Tests» і «TestResults», із відповідними зв’язками, встановленими за допомогою зовнішніх ключів. Первинні ключі забезпечують унікальну ідентифікацію кожного запису, а обмеження та індексування підтримують ефективне надсилання запитів і цілісність даних.

На додаток до основних таблиць, система також містить допоміжні структури для автентифікації користувачів, контролю доступу, журналів

аудиту та реєстрації даних для підвищення безпеки та відстеження. Реалізація наголошує на нормалізованих структурах даних, щоб мінімізувати надмірність і підтримувати узгодженість між записами.

Щоб забезпечити зручність обслуговування та масштабованість у майбутньому, базу даних розроблено з використанням модульних принципів, що передбачає потенційні вдосконалення, такі як розширені звіти, автоматичні сповіщення та інтеграція із зовнішнім лабораторним обладнанням або системами державного регулювання. Регулярне резервне копіювання, дозволи на основі ролей і узгодженість транзакцій також інтегровані в конфігурацію СУБД, щоб забезпечити надійне та безпечне операційне середовище.

Вибір бази даних є критично важливим етапом у розробці будь-якої інформаційної системи, оскільки від цього залежить не лише зберігання даних, але й загальна продуктивність, безпека та можливість масштабування системи в майбутньому. У випадку реалізації інформаційної системи для лабораторного сортового контролю було вирішено використовувати MySQL як основний засіб управління базою даних. Це рішення базувалося на ряді важливих факторів, які підтверджують доцільність такого вибору.

По-перше, MySQL є відомою реляційною системою управління базами даних з відкритим вихідним кодом, яка зарекомендувала себе як надійне та стабільне рішення для обробки великих обсягів даних. Завдяки своїй популярності MySQL має велику спільноту користувачів і розробників, що забезпечує доступ до багатьох ресурсів, документації, навчальних матеріалів та підтримки. Це особливо важливо для розробників, які можуть швидко знайти рішення для можливих проблем або отримати допомогу при налаштуванні системи [11].

По-друге, MySQL забезпечує високу продуктивність та швидкість роботи, що критично важливо для лабораторних систем, де час виконання запитів може впливати на загальну ефективність процесів. З можливістю обробки складних SQL-запитів та оптимізації індексів, MySQL гарантує

швидкий доступ до даних, що необхідно для оперативного виконання лабораторних досліджень.

По-третє, система добре інтегрується з популярними мовами програмування, такими як PHP, що було особливо важливим при розробці інформаційної системи, яка використовує фреймворк Symfony. Це забезпечує простоту у створенні запитів до бази даних та обробці отриманих результатів, що значно полегшує процес розробки та підтримки програми.

Вибір MySQL також був обумовлений його функціями забезпечення безпеки. База даних підтримує механізми управління доступом, що дозволяє реалізувати рольове управління, коли різні користувачі системи мають різні рівні доступу до даних. Це особливо важливо у лабораторному середовищі, де обробляється чутлива інформація про насіннєві зразки, результати тестування та інші критично важливі дані [11].

Крім того, MySQL пропонує масштабованість, що дозволяє системі рости разом із збільшенням обсягу даних та кількості користувачів. Це означає, що у випадку розширення функціональності системи чи збільшення навантаження, не доведеться кардинально змінювати структуру бази даних або шукати нову систему управління базами даних. MySQL може легко впоратися з підвищеним навантаженням, що робить його оптимальним вибором для майбутнього розвитку проєкту.

Зрештою, рішення про вибір MySQL для інформаційної системи лабораторного сортового контролю було ухвалено на основі його надійності, продуктивності, зручності інтеграції та масштабованості. Це забезпечить ефективну роботу системи, яка буде відповідати вимогам лабораторії та користувачів, і дозволить зберігати та обробляти важливі дані з максимальною ефективністю.

Підсумовуючи, MySQL пропонує збалансоване поєднання функціональності, продуктивності та гнучкості, що робить його оптимальним вибором для реалізації компонента бази даних інформаційної системи лабораторного контролю сортів.

Отриману схему бази даних зображено на малюнку 3.2.

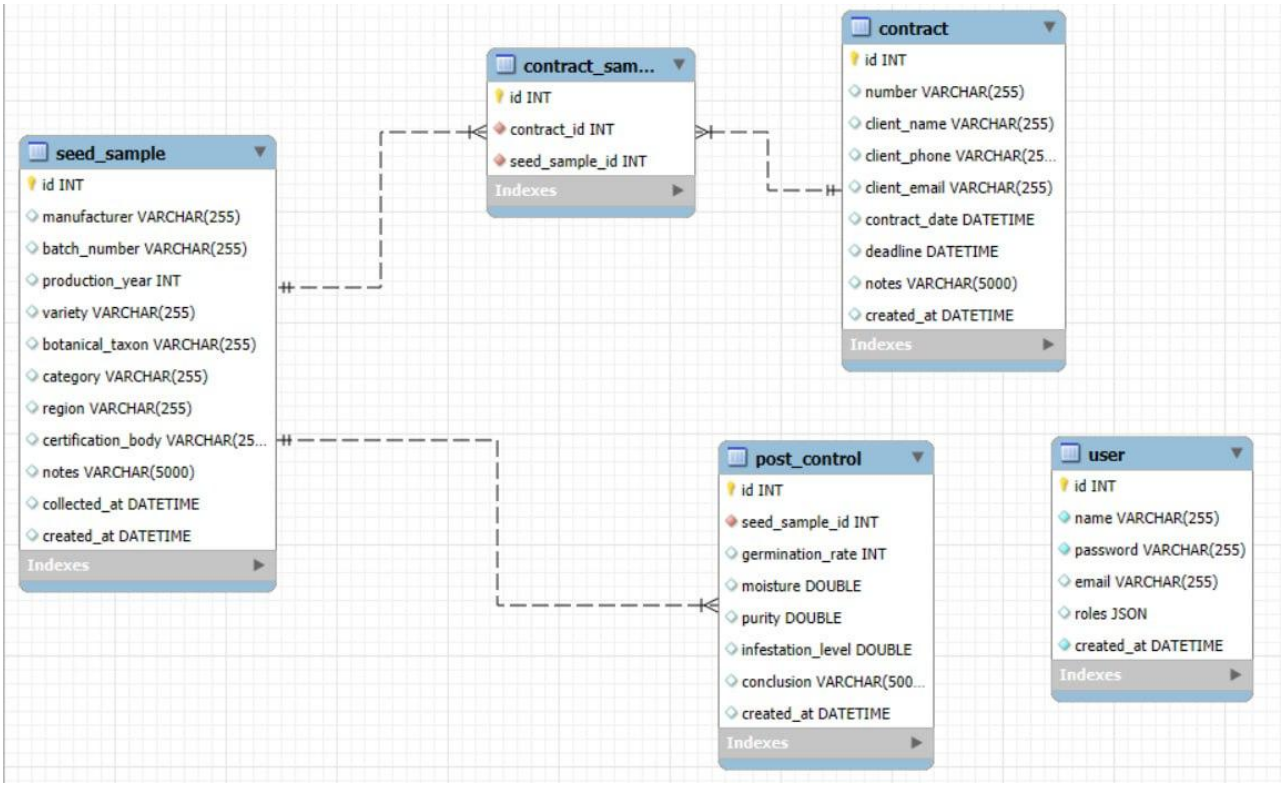

Рис. 3.2 – База даних системи

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення є ключовим аспектом розробки програмного забезпечення, що охоплює фундаментальні структури програмної системи та методології, що використовуються для їх створення. Він діє як план, який описує, як різні компоненти системи взаємодіють, потік даних та інтеграцію різних елементів для задоволення бізнес-вимог. У контексті інформаційної системи, розробленої для проведення лабораторного різноманітного контролю, ефективна архітектура програмного забезпечення має важливе значення для забезпечення масштабованості, ремонтопридатності та надійності системи, що дозволяє їй ефективно справлятися зі складністю лабораторних операцій [12].

Архітектуру системи лабораторного сортоконтролю можна концептуалізувати в кілька рівнів, кожен з яких виконує окрему функцію. Рівень презентації формує інтерфейс користувача, де відбувається взаємодія між користувачами та системою. Цей рівень включає веб-сторінки або форми, які дозволяють лаборантам, клієнтам і адміністраторам отримувати доступ до даних і керувати ними. Він розроблений так, щоб бути інтуїтивно зрозумілим і зручним для користувача, гарантуючи, що користувачі можуть легко орієнтуватися в системі під час виконання таких завдань, як додавання зразків, проведення тестів і створення звітів.

Під рівнем презентації знаходиться прикладний рівень, також відомий як рівень бізнес-логіки. Цей компонент відповідає за обробку запитів користувачів і виконання основних функцій системи. Він охоплює бізнес-правила, які визначають, як дані створюються, читаються, оновлюються та видаляються, ефективно керуючи взаємодією між рівнями презентації та даних, щоб забезпечити правильне виконання операцій відповідно до встановленої логіки.

Рівень даних призначений для зберігання та пошуку даних, надаючи важливий інтерфейс до системи керування базами даних, якою в даному випадку є MySQL. Цей рівень керує збереженням даних, пов’язаних із зразками насіння, контрактами, результатами випробувань та інформацією про користувачів. Абстрагуючи складність управління даними від прикладного рівня, це дозволяє розробникам зосередитися на реалізації бізнес-логіки, не обтяжуючись тонкощами операцій з базою даних.

У сучасній архітектурі програмного забезпечення інтеграційний рівень має важливе значення для полегшення зв’язку між лабораторною системою контролю різноманітності та зовнішніми системами чи службами. Цей рівень забезпечує безперебійний потік даних за допомогою API (інтерфейсів прикладного програмування) і веб-сервісів, дозволяючи

інтеграцію з лабораторним обладнанням сторонніх виробників, нормативними базами даних і інструментами звітності. Це підключення покращує загальну функціональність і взаємодію системи.

Безпека є критично важливим питанням в архітектурі програмного забезпечення, особливо для системи, що керує конфіденційною інформацією. Рівень безпеки охоплює механізми автентифікації та авторизації, призначені для захисту даних і забезпечення доступу до певних функцій лише авторизованим користувачам. Для захисту конфіденційної інформації впроваджено такі заходи, як шифрування даних, безпечні процеси автентифікації користувачів і контроль доступу на основі ролей.

Крім того, архітектурний проєкт інформаційної системи повинен враховувати нефункціональні вимоги, такі як продуктивність, масштабованість, зручність обслуговування та надійність. Ефективна архітектура повинна передбачати майбутні вдосконалення та враховувати зростаючі обсяги даних у міру розширення лабораторії. Використовуючи такі шаблони проєктування, як MVC (Model-View-Controller), розробники можуть досягти чіткого розподілу завдань, спрощуючи керування та сприяючи еволюції системи з часом [13].

Підсумовуючи, архітектура програмного забезпечення є життєво важливою складовою інформаційної системи для проведення лабораторного сортоконтролю. Ретельно проєктуючи структуру системи на різних рівнях, розробники можуть створити масштабовану, ефективну та безпечну програму, яка відповідає потребам користувачів і підтримує складні процеси, властиві лабораторним операціям.

Нижче буде продемонстровано багатошарова архітектура даного програмного забезпечення (рис. 3.3)

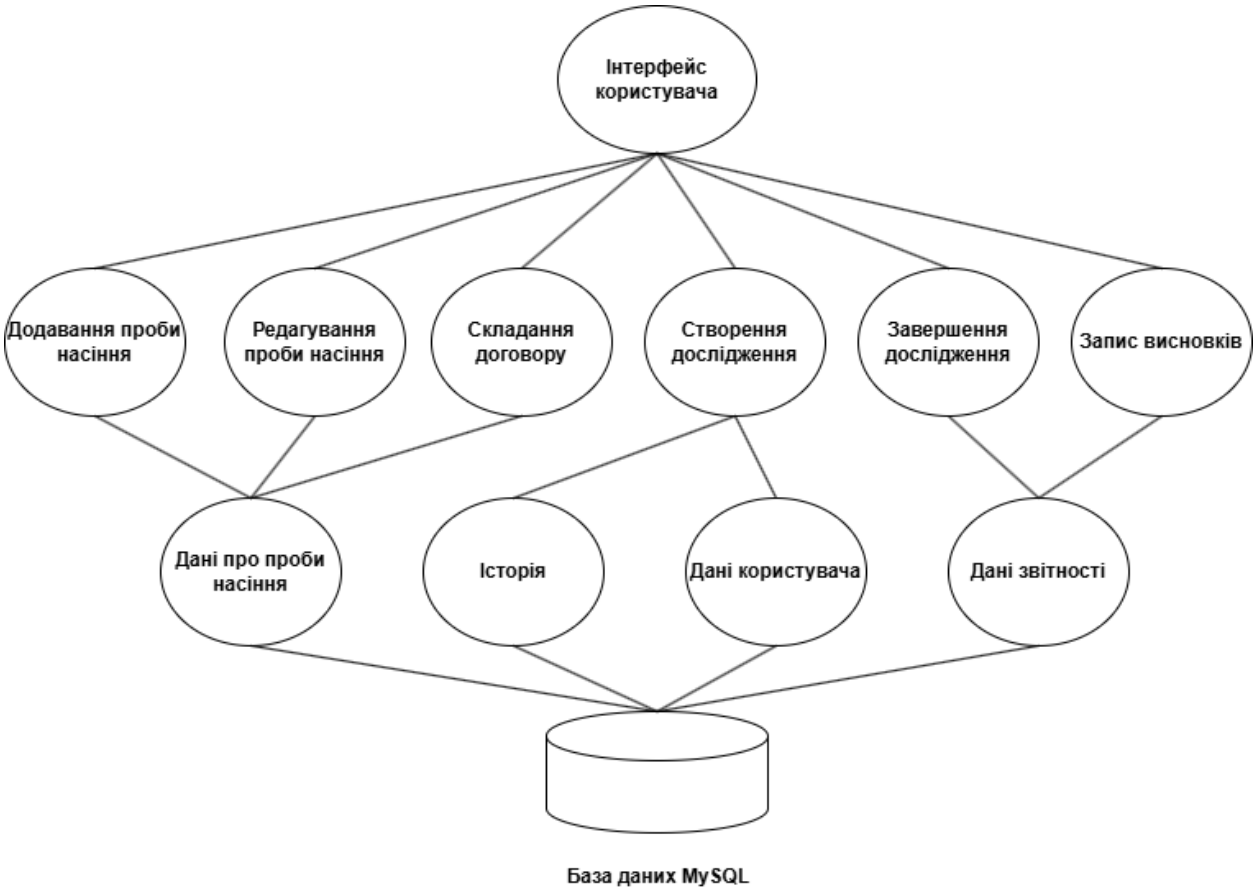

Рис. 3.3 – Багатошарова архітектура веб-сайту

Чотири-рівнева архітектура інформаційної системи для проведення лабораторного сортоконтролю включає презентаційний рівень, прикладний рівень, рівень даних та рівень інтеграції. Презентаційний рівень забезпечує інтуїтивно зрозумілий інтерфейс користувача для лаборантів, клієнтів і адміністраторів для взаємодії з системою, забезпечуючи легку навігацію та виконання завдань. Прикладний рівень містить основну бізнес-логіку, обробку введених даних користувача та керування взаємодією між рівнями презентації та даних. Рівень даних відповідає за зберігання та отримання інформації, взаємодіючи з базою даних MySQL для забезпечення ефективного керування даними. Рівень бази даних полегшує зв’язок із зовнішніми системами та службами через API, забезпечуючи безперебійний обмін даними з лабораторним обладнанням сторонніх виробників і нормативними базами даних. Така архітектурна конструкція гарантує, що система є масштабованою,

зручною в обслуговуванні та здатною відповідати складним вимогам лабораторних операцій.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Пакетна діаграма — це тип структурної діаграми в уніфікованій мові моделювання (UML), яка візуально представляє організацію системи в пакети, показуючи, як ці пакети взаємодіють один з одним. Він забезпечує високорівневе уявлення про архітектуру системи, допомагаючи керувати складністю шляхом групування пов’язаних класів, компонентів або підсистем у пакети. Така організація підвищує модульність і робить систему легшою для розуміння та підтримки [14].

Основна мета діаграми пакетів полягає в тому, щоб проілюструвати залежності та зв’язки між різними пакетами в програмній системі. Кожен пакет представлений у вигляді прямокутної коробки, яка може містити кілька класів або компонентів. Діаграма містить стрілки, що вказують на залежності, де один пакет покладається на інший для правильного функціонування. Наприклад, якщо пакет A залежить від класів із пакета B, спрямована стрілка малюється від пакета A до пакета B, що означає, що зміни в пакеті B можуть вплинути на пакет A.

Діаграми пакетів особливо корисні у великих системах, оскільки вони допомагають розробникам і зацікавленим сторонам візуалізувати організацію коду, сприяють повторному використанню та сприяють кращому управлінню проєктами. Забезпечуючи чітке уявлення про модульну структуру системи, діаграми пакетів допомагають визначити межі функціональності, сприяючи поділу завдань і покращуючи співпрацю між командами розробників.

Таким чином, пакетна діаграма є важливим інструментом в архітектурі програмного забезпечення, забезпечуючи ефективну організацію та керування складними системами. Це допомагає зрозуміти високорівневу

структуру програмного забезпечення, гарантуючи, що розробники можуть орієнтуватися та ефективно підтримувати кодову базу.

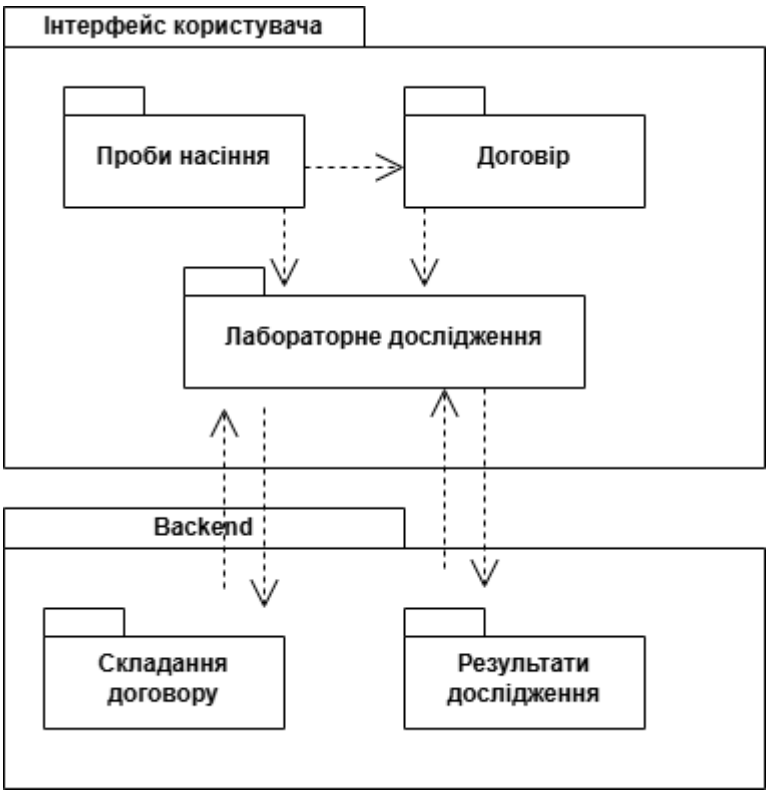

Рис. 3.4 – Діаграма пакетів

3.5 Вибір інструментарію для створення програмного забезпечення

Розробка інформаційної системи для лабораторного сортового контролю спирається на надійний інструментарій, який поєднує різні технології для забезпечення ефективної функціональності, масштабованості та зручності обслуговування. Вибрані інструменти включають PHP, Symfony, MySQL, Doctrine ORM, MySQL Workbench, HTML, CSS і JavaScript, кожен з яких служить певній меті в загальній архітектурі системи.

PHP є основною серверною мовою сценаріїв, яка використовується в проєкті. Відомий своєю гнучкістю та простотою інтеграції з різними системами баз даних, PHP особливо добре підходить для веб-розробки. Його широке використання та широка підтримка спільноти дозволяють

розробникам використовувати численні бібліотеки та фреймворки, прискорюючи процес розробки.

Symfony — це потужна структура PHP, яка спрощує розробку складних веб-додатків. Він пропагує найкращі практики розробки програмного забезпечення та надає багаторазові компоненти, які сприяють швидкій розробці програм. Завдяки таким функціям, як маршрутизація, створення шаблонів і безпека, Symfony допомагає розробникам підтримувати чисту та організовану кодову базу, підвищуючи загальну ефективність проєкту.

Для зберігання та керування даними MySQL було обрано як систему керування реляційною базою даних. Він пропонує надійність, продуктивність і надійні можливості обробки даних. MySQL добре підходить для додатків, які вимагають структурованого зберігання даних, що робить його ідеальним вибором для керування інформацією, пов’язаною зі зразками насіння, контрактами, результатами випробувань і обліковими записами користувачів у системі контролю сортів лабораторії.

Doctrine ORM — це інструмент об’єктно-реляційного відображення (ORM), який бездоганно інтегрується з Symfony, забезпечуючи рівень абстракції між додатком і базою даних. Він спрощує взаємодію з базою даних, дозволяючи розробникам працювати з об’єктами PHP замість написання необроблених запитів SQL. Це покращує читабельність коду, зручність обслуговування та гнучкість під час модифікації схеми чи логіки бази даних.

Для візуального проєктування та керування базою даних MySQL використовується MySQL Workbench. Цей інструмент надає графічний інтерфейс для моделювання бази даних, що дозволяє розробникам створювати, змінювати та ефективно керувати схемами бази даних. MySQL Workbench також полегшує створення сценаріїв SQL і допомагає виконувати завдання адміністрування бази даних, полегшуючи підтримку цілісності та структури даних.

Для презентаційного рівня програми використовуються HTML, CSS і JavaScript. HTML утворює основу веб-сторінок, структуруючи вміст і елементи для взаємодії з користувачем. CSS використовується для стилізації програми, гарантуючи, що інтерфейс користувача є візуально привабливим і відповідає бажаному бренду. JavaScript покращує інтерактивність програми, уможливлюючи такі динамічні функції, як перевірка форм, маніпулювання даними та оновлення в реальному часі, тим самим покращуючи загальну взаємодію з користувачем.

Підсумовуючи, вибраний інструментарій для інформаційної системи лабораторного сортового контролю охоплює комбінацію мов програмування, фреймворків та інструментів керування базами даних. Використовуючи PHP, Symfony, MySQL, Doctrine ORM, MySQL Workbench, HTML, CSS і JavaScript, команда розробників може створити надійну, масштабовану та зручну програму, яка відповідає потребам лабораторних операцій і покращує керування процесами сортовипробування.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для забезпечення ефективної роботи інформаційної системи для проведення лабораторного сортоконтролю важливо відповідати спеціальним апаратним і програмним вимогам, які забезпечать продуктивність, надійність і масштабованість системи.

Починаючи з апаратного забезпечення, сервер повинен бути оснащений багатоядерним процесором, таким як Intel Xeon або AMD Ryzen, що містить мінімум чотири ядра. Ця можливість має вирішальне значення для ефективної обробки кількох запитів і одночасних користувачів. Сервер також повинен мати принаймні 16 ГБ оперативної пам’яті, що сприятиме безперебійній роботі додатків і баз даних, особливо під час пікового використання. Також необхідний відповідний обсяг пам’яті, принаймні 500 ГБ твердотільного накопичувача, рекомендованого для розміщення операційної системи, файлів програми та бази даних MySQL. Використання SSD є кращим через їх швидший доступ до даних і покращену продуктивність порівняно з традиційними жорсткими дисками. Крім того, для забезпечення стабільних і швидких з’єднань для передачі даних між сервером і клієнтськими пристроями потрібна надійна мережева інтерфейсна карта Ethernet (NIC).

Клієнтські робочі станції також повинні відповідати певним критеріям для забезпечення оптимальної продуктивності. Кожен комп’ютер або ноутбук повинен мати принаймні 8 ГБ оперативної пам’яті та двоядерний процесор для роботи веб-браузерів і ефективного доступу до програми. Монітор з роздільною здатністю не менше 1920x1080 пікселів забезпечить чітку видимість інтерфейсу програми та звітів. Стандартні периферійні пристрої,

такі як клавіатури та миші, необхідні, а додаткові пристрої, такі як сканери штрих-кодів, можуть підвищити швидкість введення зразкової інформації.

Мережеве обладнання відіграє важливу роль у полегшенні зв'язку всередині системи. Високошвидкісний маршрутизатор необхідний для з'єднання серверних і клієнтських пристроїв, тоді як комутатор може знадобитися в налаштуваннях локальної мережі (LAN) для підключення кількох клієнтів до сервера.

З боку програмного забезпечення сервер повинен працювати на сумісній системі, такій як Ubuntu Server або Windows Server, для розміщення програми та бази даних. Клієнтські машини можуть використовувати будь-яку сучасну операційну систему, включаючи Windows, macOS або Linux, забезпечуючи сумісність із веб-браузерами.

Для обслуговування веб-програми та керування запитами потрібен веб-сервер, наприклад Apache або Nginx. Система управління реляційною базою даних MySQL буде встановлена на сервері для управління зберіганням і пошуком даних, а платформа Symfony буде використана для розробки веб-додатку, забезпечуючи структуроване середовище, яке сприяє ефективній розробці програмного забезпечення.

Основною мовою сценаріїв на стороні сервера, яка використовується в цьому проєкті, буде PHP, що дозволить обробляти логіку програми та взаємодію з сервером. Для розробки інтерфейсу використовуватимуться HTML, CSS і JavaScript, щоб створити привабливий і зручний інтерфейс.

Для спрощення взаємодії з базою даних буде інтегровано Doctrine ORM, що дозволить розробникам працювати з об'єктами PHP замість необроблених запитів SQL. Крім того, MySQL Workbench буде використовуватися як графічний інструмент для проєктування та керування базою даних MySQL, полегшуючи створення схем і візуалізацію даних.

Нарешті, система контролю версій, така як Git, буде реалізована для керування змінами коду та підтримки співпраці між розробниками.

Інтегроване середовище розробки (IDE), наприклад PhpStorm або Visual Studio Code, підвищить продуктивність у процесі розробки.

Таким чином, вимоги до апаратного та програмного забезпечення для інформаційної системи, розробленої для лабораторного контролю сортів, є вирішальними для забезпечення оптимальної продуктивності, безпеки та взаємодії з користувачем. Виконання цих вимог дозволить системі ефективно справлятися зі складнощами, пов’язаними з лабораторними операціями.

На рисунку 4.1 зображено діаграму розгортання даної системи. Клієнт-серверна архітектура реалізована на двох окремих серверах.

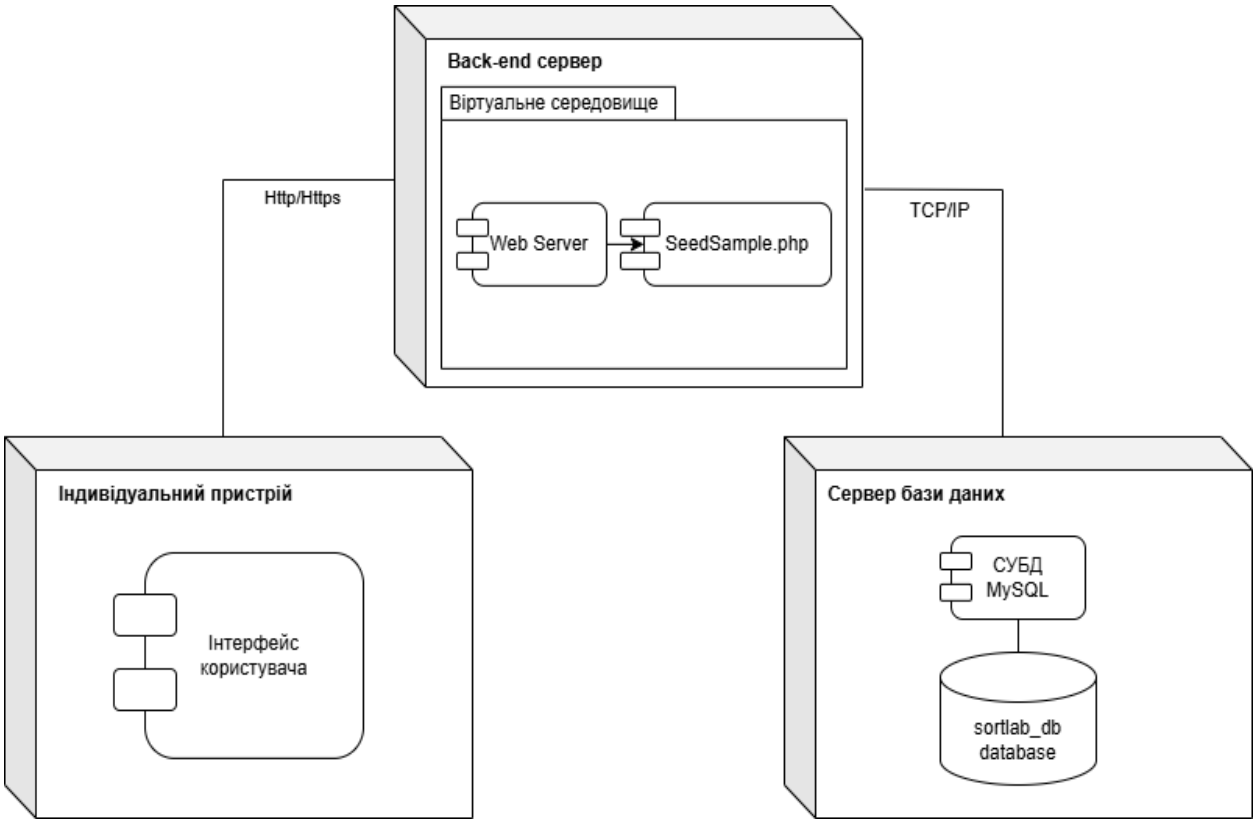

Рис. 4.1 – Діаграма розгортання

4.2 Тестування системи

Запустивши систему, бачимо головну сторінку програмного забезпечення (рис. 4.2).

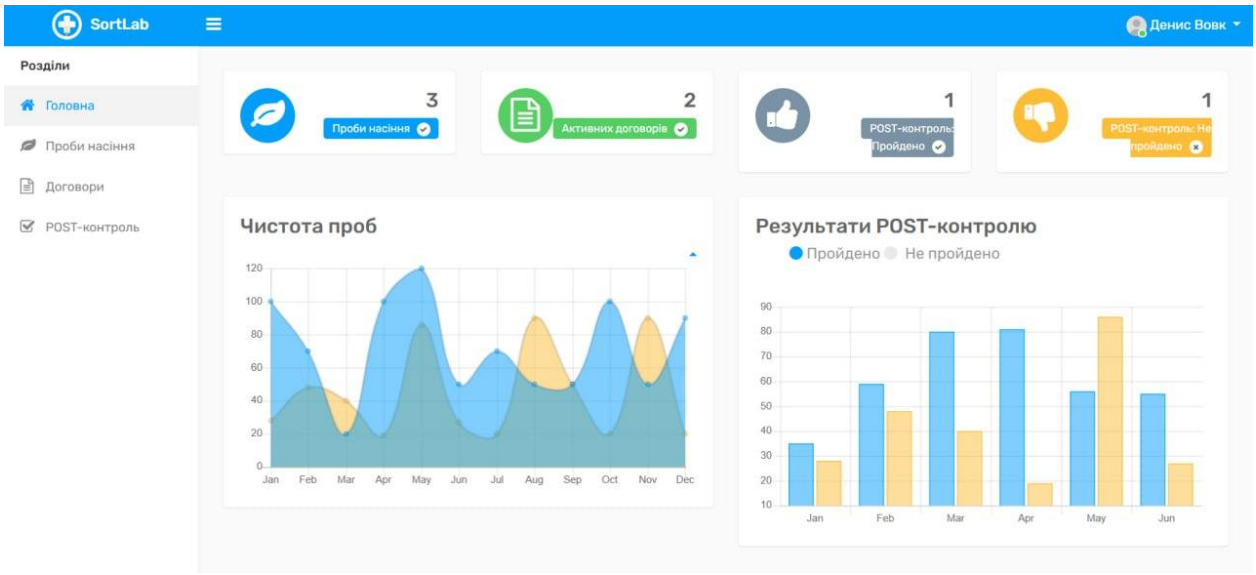

Рис. 4.2 – Головна сторінка

На головній сторінці нашої програми ми бачимо інформаційну панель. На ній у нас представлені деякі статистичні дані на кшталт кількості проб насіння, договорів та результатів досліджень з їх оцінкою. Також тут є графіки для візуального подання інформації працівникові.

Тепер відкриваємо сторінку "Проби насіння", тут ми маємо список вже доданих проб насіння з усією необхідною інформацією. Також ми можемо створити нову пробу, натиснувши на відповідну кнопку (рис. 4.3).

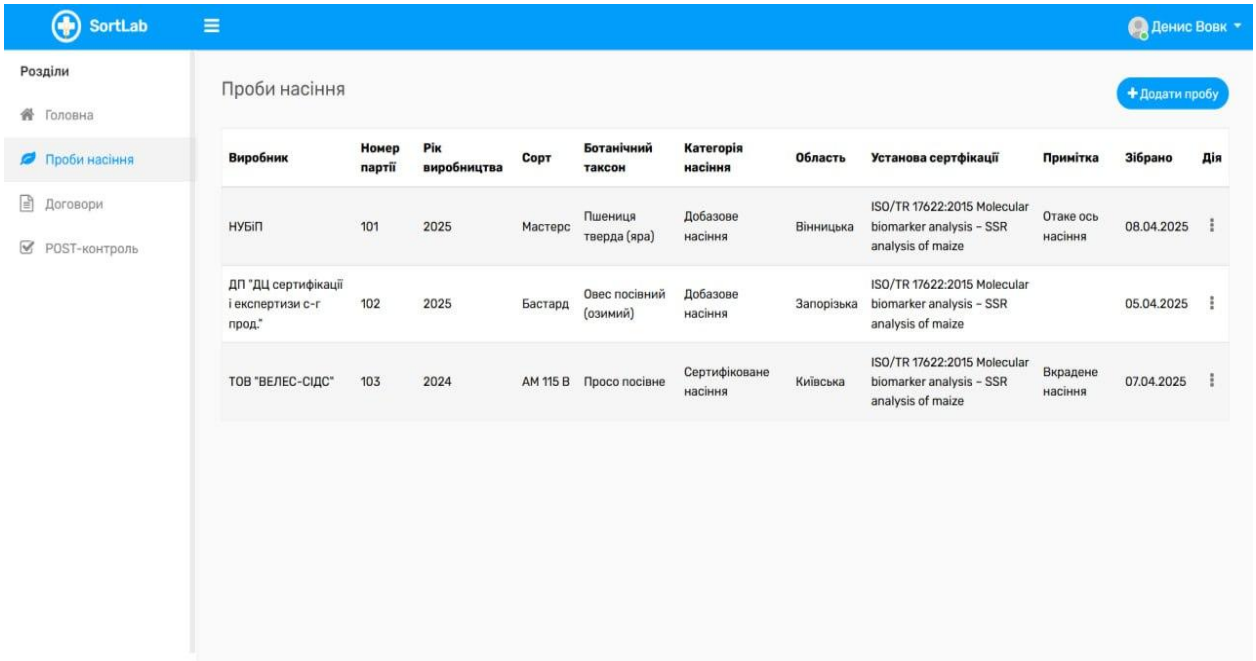

Рис. 4.3 – Сторінка “Проби насіння”

На сторінку створення проби насіння ми маємо всі потрібні поля для заповнення. Після того як користувач введе дані і натисне кнопку підтвердження, нова проба насіння тут же буде записана в базу даних (рис. 4.4).

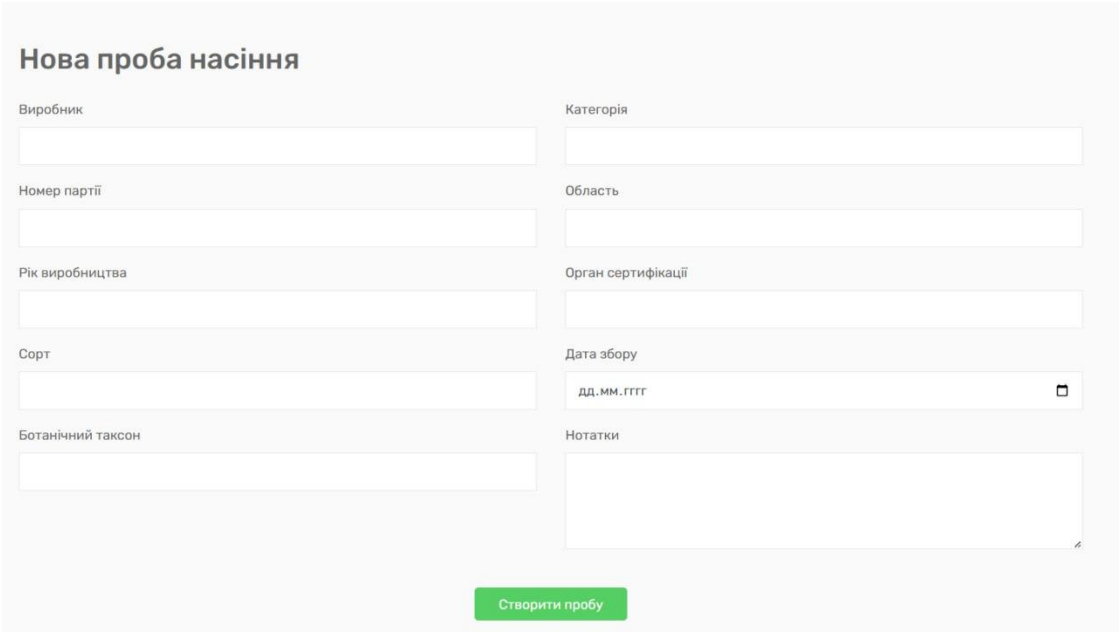

Рис. 4.4 – Форма створення проби насіння

Тепер відкриємо сторінку "Договори". Тут перед нами представлені списки активних договорів. Ми можемо створити новий договір, натиснувши кнопку створення в правому верхньому кутку екрана (рис. 4.5).

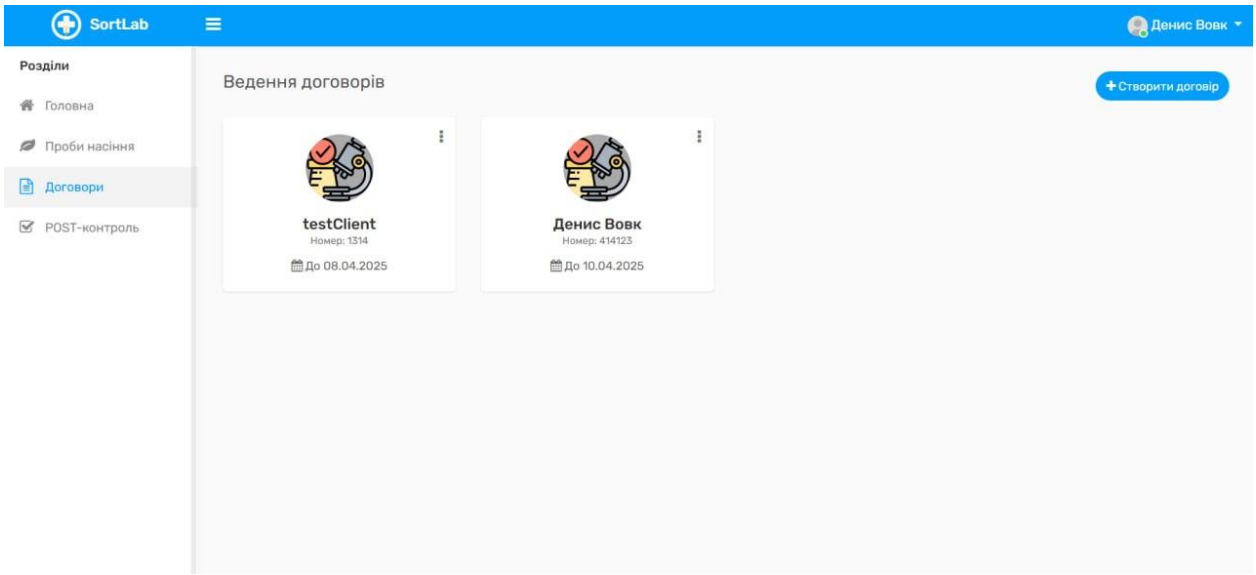

Рис. 4.5 – Сторінка “Договори”

За таким же принципом вводимо дані і створюємо новий договір за допомогою списку з пробами насіння (рис. 4.6).

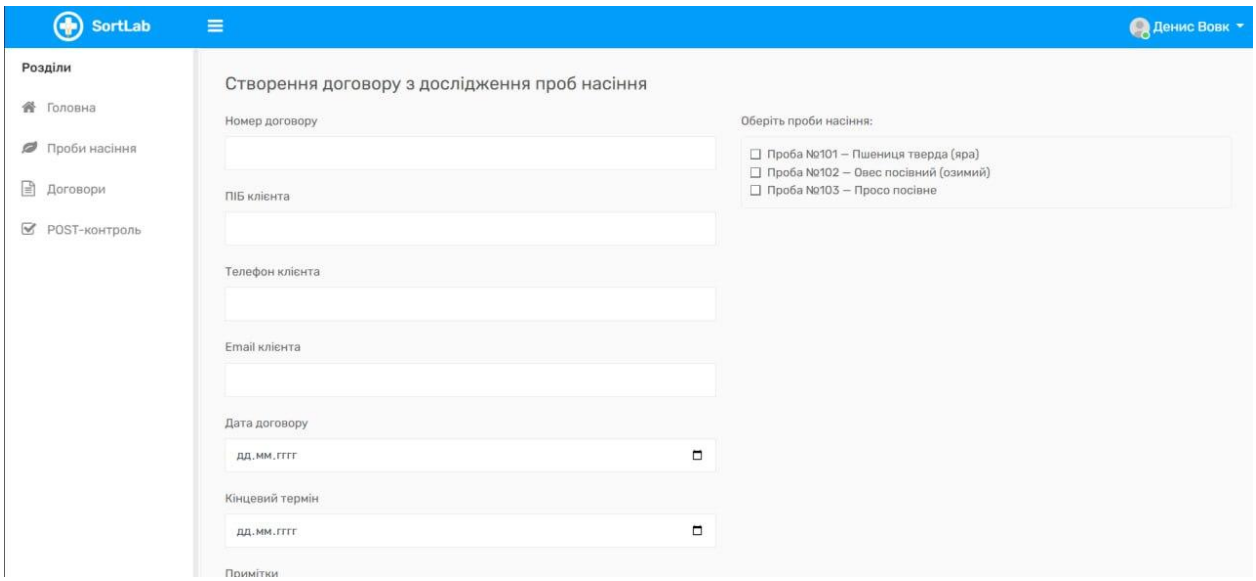

Рис. 4.6 – Форма створення договору

Відкриваємо сторінку Пост-контроль. Тут у нас представлені дослідження, що проводяться лаборантами. У кожного дослідження тут є номер договору, проба насіння та результат (рис. 4.7).

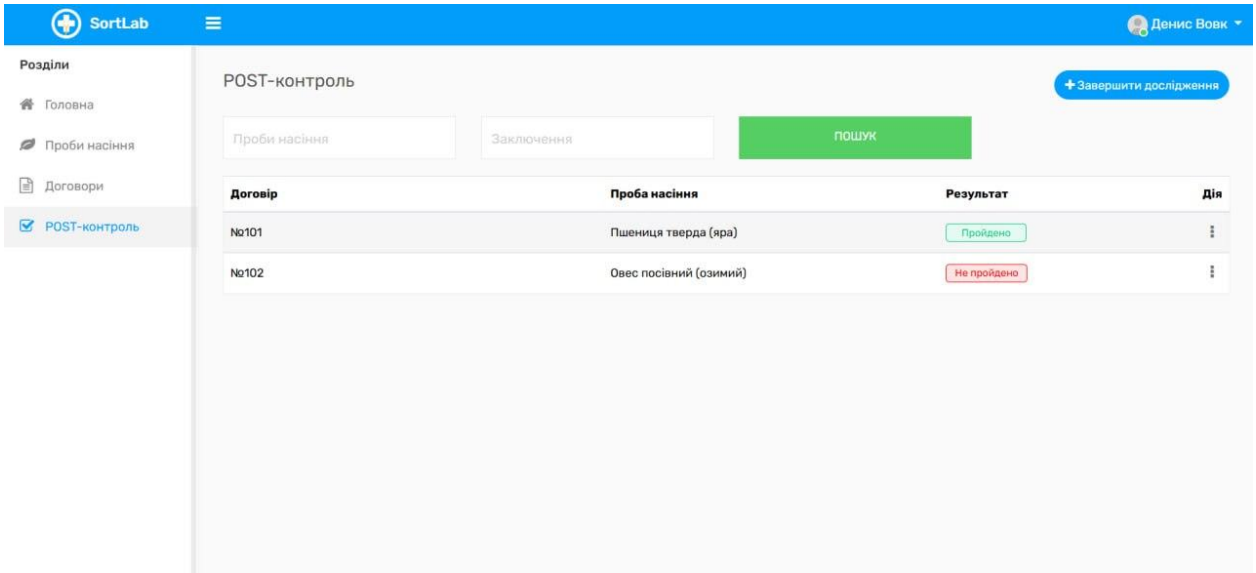

Рис. 4.7 – Сторінка “POST-контроль”

Ще одна форма, тільки вже щодо створення результатів дослідження. Тут лаборант заповнює всі показники після проведеного дослідження та вказує

фінальний висновок (рис. 4.8).

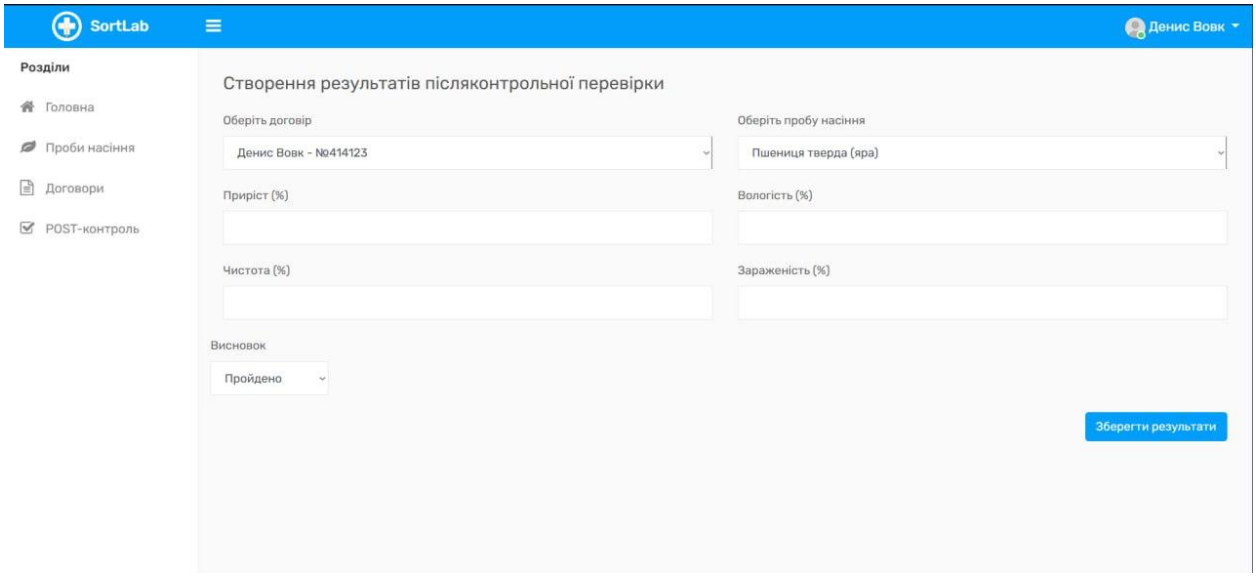

Рис. 4.8 – Форма створення результатів пост-контролю

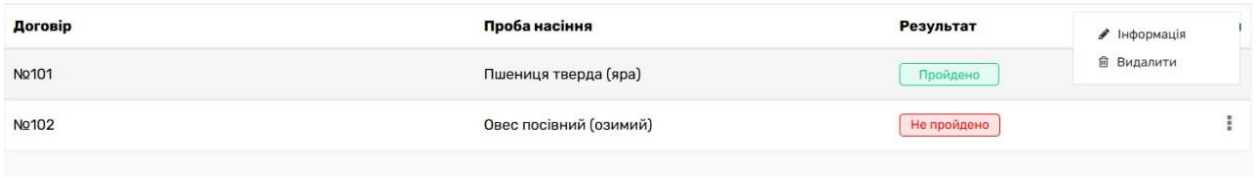

Рис. 4.9 – Модальне вікно

Натиснувши на 3 точки праворуч від дослідження, у нас відкриється модальне вікно, що управляє, з можливістю подивитися інформацію або видалити дослідження зовсім (рис. 4.10).

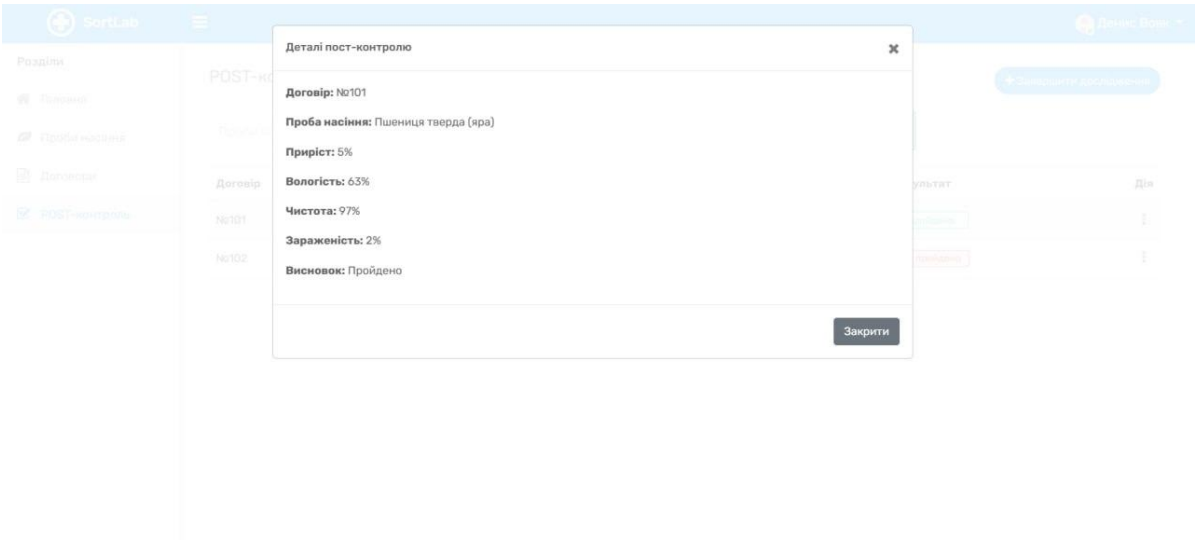

Рис. 4.10 – Інформація про дослідження

І так само, у цьому додатку є авторизація користувача. Тут у нас представлена форма входу в систему за допомогою логіну та пароля (рис. 4.11).

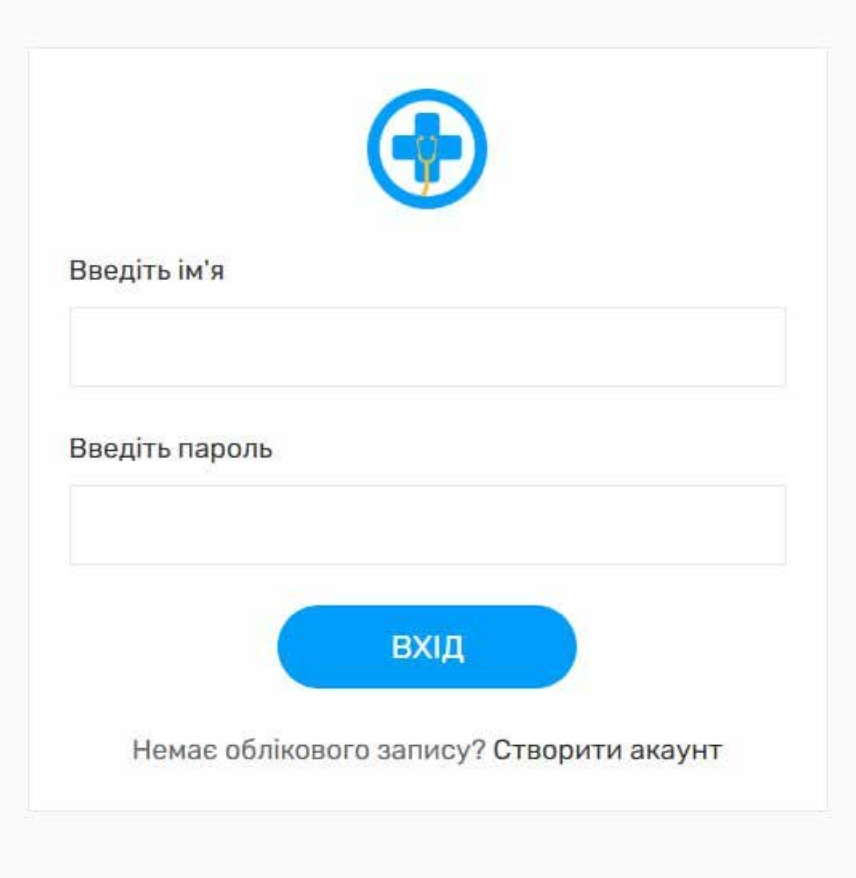

Рис. 4.11 – Форма авторизації

ВИСНОВКИ

Підсумовуючи, розробка інформаційної системи для проведення лабораторного контролю сортів є значним прогресом в управлінні якістю насіння та дотриманням нормативних вимог у сільськогосподарській практиці. Ця система розроблена для оптимізації роботи лабораторії, підвищення точності даних і покращення користувацького досвіду для всіх зацікавлених сторін, включаючи лаборантів, клієнтів і адміністративний персонал.

Завдяки інтеграції сучасних технологій, таких як PHP, Symfony і MySQL, система забезпечує надійну структуру, яка підтримує ефективну обробку початкових зразків, процеси тестування та документацію результатів. Чотирирівнева архітектура забезпечує чіткий розподіл проблем, сприяючи масштабованості та зручності обслуговування, одночасно сприяючи бездоганній взаємодії між користувачами та системою. Застосовуючи логічну модель даних і використовуючи такі інструменти, як Doctrine ORM, архітектура забезпечує ефективне керування даними та їх пошук, гарантуючи, що життєво важлива інформація завжди доступна та надійна.

Ретельний вибір апаратних і програмних компонентів сприяє продуктивності та безпеці системи, дозволяючи їй відповідати суворим вимогам лабораторних операцій. Крім того, впровадження заходів безпеки та керування ролями користувачів захищає конфіденційні дані, забезпечуючи відповідність галузевим нормам.

Ця інформаційна система не тільки вирішує поточні виклики сортовипробування, але й дає можливість лабораторіям адаптуватися до майбутніх досягнень технологій і методологій. Модульна конструкція дозволяє легко вдосконалювати та інтегрувати із зовнішніми системами, гарантуючи, що система залишається актуальною та ефективною в міру розвитку сільськогосподарського ландшафту.

Впровадження інформаційної системи призведе до значного підвищення ефективності роботи в лабораторних середовищах. Завдяки автоматизації рутинних завдань, таких як реєстрація зразків, документування результатів і звітність, персонал лабораторії може зосередитися на більш цінних видах діяльності, таких як аналіз даних і вдосконалення методологій тестування. Зручний інтерфейс системи дозволяє користувачам швидко переміщатися між різними функціями, тим самим скорочуючи час навчання та підвищуючи загальну продуктивність. Крім того, включення функцій аналітики даних дозволить керівникам лабораторій відстежувати показники ефективності, визначати тенденції в якості насіння та приймати обґрунтовані рішення на основі даних у реальному часі. Ця можливість не тільки оптимізує внутрішні процеси, але й покращує реагування лабораторії на потреби клієнтів і нормативні вимоги.

Загалом, успішне впровадження цієї інформаційної системи закладає надійну основу для покращення процесів забезпечення якості в лабораторіях випробування насіння. Сприяючи кращому управлінню даними, покращуючи зв’язок між зацікавленими сторонами та забезпечуючи дотримання нормативних стандартів, система в кінцевому підсумку сприяє надійності та якості сільськогосподарської продукції. Дослідження та розробки, проведені протягом цього проєкту, дали цінну інформацію та практичне рішення, яке може суттєво вплинути на сферу лабораторного контролю сортів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SeedManager – [Електронний ресурс] – Режим доступу: https://manager.getseed.io/users/sign_in
2. LabWare LIMS – [Електронний ресурс] – Режим доступу: https://www.labware.com/industries/process-chemical
3. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ru/glossary/mnogourovnevaya-arhitektura/
4. The Clean Architecture – [Електронний ресурс] – Режим доступу: https://medium.com/clean-code-channel/clean-architecture-the-solution-to-have-a-reusable-flexible-and-testable-code-ac7e296d1a75
5. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
6. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
7. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/
8. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html
9. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

10. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.
11. W3Schools – [Електронний ресурс] – Режим доступу: www.w3schools.com
12. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
13. What is Component Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
14. Deployment Diagram in UML: Definition, Examples & Components – [Електронний ресурс] – Режим доступу: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html
15. What is a data flow diagram? – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram
16. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
17. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
18. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html
19. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ru/uml-classes/
20. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html

21. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
22. Лі, С. (2020). «Схеми для електронних книг: погляд на схожість і поведінку читачів». Дослідження в галузі бібліотечної та інформаційної науки, 42 (2), 135-150.
23. Zhou, Y., & Zhang, X. (2019). «Управління даними в сільськогосподарських дослідженнях: найкращі практики та нові технології». Journal of Agricultural Science and Technology, 21(4), 45-60.
24. Сміт, Дж. Р., і Джонсон, Л. М. (2018). «Впровадження інформаційних систем у лабораторних середовищах: виклики та рішення». Міжнародний журнал управління інформацією, 38 (3), 234-245.
25. Ван, Т., і Лю, Х. (2021). «Роль технологій у підвищенні ефективності лабораторії та точності даних». Журнал лабораторної автоматизації, 26 (1), 11-23.
26. Гарсія Р. та Лопес М. (2019). «Найкращі практики управління якістю насіння в сільськогосподарських лабораторіях». Crop Science Journal, 59(2), 511-520.
27. Патель, В., і Кумар, Р. (2020). «Веб-інформаційні системи для управління сільськогосподарськими даними: огляд». Journal of Computer and Agricultural Engineering, 10(2), 90-104.
28. Бейкер, А., Томас, С. (2017). «Вплив інформаційних технологій на сільськогосподарську практику». Дослідження сільськогосподарської інформації, 15 (1), 20-30.
29. О'Брайен, Т. (2018). «Проєктування зручних інтерфейсів для лабораторних інформаційних систем». Journal of Usability Studies, 13(4), 185-200.
30. Тернер, П., Хейз, К. (2021). «Майбутнє сільськогосподарських лабораторій: інтеграція IoT та аналітики даних». Journal of Smart Agriculture, 5(1), 15-29.

31. Нгуєн, Л. Т. та Кім, С. Дж. (2020). «Ефективність використання систем управління базами даних у лабораторних роботах». Журнал управління базами даних, 31(2), 1-16.
32. Робертс К. та Еванс Т. (2019). «Покращення якості тестування насіння за допомогою інформаційно-технологічних рішень». Journal of Agricultural Technology, 12(3), 255-267.
33. Грін, Д., і Філд, Дж. (2020). «Цілісність даних і керування ними в лабораторних умовах: практичне дослідження». Міжнародний журнал лабораторних досліджень, 18 (1), 45-58.
34. Чжао, К., і Ван, С. (2021). «Застосування хмарних обчислень в управлінні сільськогосподарськими даними». Journal of Cloud Computing: Advances, Systems and Applications, 10(2), 120-135.
35. Елліс Р. та Андерсон М. (2018). «Принципи проєктування систем управління лабораторною інформацією». Журнал інженерії інформаційних систем, 14 (2), 75-89.

ДОДАТОК А

Фрагменти програмного коду. Функція додання проби насіння

```
class SeedController extends AbstractController
{
    #[Route('/seeds', name: 'app_seed')]
    public function index(SeedSampleRepository $seedRepository): Response
    {
        $seeds = $seedRepository->findAll();

        return $this->render('seed/index.html.twig', [
            'seeds' => $seeds
        ]);
    }

    #[Route('/seed/create', name: 'seed_create', methods: ['GET', 'POST'])]
    public function create(Request $request, EntityManagerInterface
$entityManager): Response
    {
        if ($request->isMethod('POST')) {
            $seed = new SeedSample();

            $seed->setManufacturer($request->request->get('manufacturer'));
            $seed->setBatchNumber($request->request->get('batchNumber'));
            $seed->setProductionYear((int)$request->request-
>get('productionYear'));
            $seed->setVariety($request->request->get('variety'));
            $seed->setBotanicalTaxon($request->request->get('botanicalTaxon'));
            $seed->setCategory($request->request->get('category'));
            $seed->setRegion($request->request->get('region'));
            $seed->setCertificationBody($request->request-
>get('certificationBody'));
            $seed->setNotes($request->request->get('notes'));
            $collectedAt = $request->request->get('collectedAt');

            if ($collectedAt) {
                $seed->setCollectedAt(new \DateTimeImmutable($collectedAt));
            }
            $seed->setCreatedAt(new \DateTimeImmutable());

            $entityManager->persist($seed);
            $entityManager->flush();

            return $this->redirectToRoute('app_seed');
        }

        return $this->render('seed/create.html.twig');
    }
```

```
    #[Route('/seeds/{id}/edit', name: 'seed_edit')]
    public function edit(int $id, Request $request, EntityManagerInterface $em,
SeedSampleRepository $seedRepository): Response
    {
        $seed = $seedRepository->find($id);

        if (!$seed) {
            throw $this->createNotFoundException('Пробу не знайдено');
        }

        if ($request->isMethod('POST')) {
            $seed->setManufacturer($request->request->get('manufacturer'));
            $seed->setBatchNumber($request->request->get('batchNumber'));
            $seed->setProductionYear((int) $request->request-
>get('productionYear'));
            $seed->setVariety($request->request->get('variety'));
            $seed->setBotanicalTaxon($request->request->get('botanicalTaxon'));
            $seed->setCategory($request->request->get('category'));
            $seed->setRegion($request->request->get('region'));
            $seed->setCertificationBody($request->request-
>get('certificationBody'));
            $seed->setNotes($request->request->get('notes'));
            $collectedAt = $request->request->get('collectedAt');
            $seed->setCollectedAt($collectedAt ? new
\DateTimeImmutable($collectedAt) : null);

            $em->flush();

            return $this->redirectToRoute('app_seed');
        }

        return $this->render('seed/edit.html.twig', [
            'seed' => $seed
        ]);
    }

    #[Route('/seeds/{id}/delete', name: 'seed_delete', methods: ['POST'])]
    public function delete(int $id, EntityManagerInterface $em,
SeedSampleRepository $seedRepository): Response
    {
        $seed = $seedRepository->find($id);

        if ($seed) {
            $em->remove($seed);
            $em->flush();
        }

        return $this->redirectToRoute('app_seed');
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Функціонал виконання дослідження

```php
<?php

namespace App\Controller;

use App\Entity\Contract;
use App\Entity\PostControl;
use App\Entity\SeedSample;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class PostControlController extends AbstractController
{
    #[Route('/post_control', name: 'app_post_control')]
    public function index(EntityManagerInterface $entityManager): Response
    {
        $controls = $entityManager->getRepository(PostControl::class)->findAll();

        return $this->render('post_control/index.html.twig', [
            'controls' => $controls,
        ]);
    }

    #[Route('/post_control/create', name: 'post_control_create')]
    public function create(Request $request, EntityManagerInterface $em):
Response
    {
        $contracts = $em->getRepository(Contract::class)->findAll();

        if ($request->isMethod('POST')) {
            $sampleId = $request->request->get('seedSample');
            dump($sampleId);
            $sample = $em->getRepository(SeedSample::class)->find($sampleId);
            dump($sample);

            $postControl = new PostControl();
            $postControl->setSeedSample($sample);
            $postControl->setGerminationRate((int)$request->request-
>get('germinationRate'));
            $postControl->setMoisture((float)$request->request->get('moisture'));
            $postControl->setPurity((float)$request->request->get('purity'));
            $postControl->setInfestationLevel((float)$request->request-
>get('infestationLevel'));
            $postControl->setConclusion($request->request->get('conclusion'));
```

```
            $postControl->setCreatedAt(new \DateTimeImmutable());

            $em->persist($postControl);
            $em->flush();

            return $this->redirectToRoute('app_post_control');
        }

        return $this->render('post_control/create.html.twig', [
            'contracts' => $contracts,
        ]);
    }

    #[Route('/post-control/delete/{id}', name: 'post_control_delete')]
    public function delete(PostControl $control, EntityManagerInterface $em):
Response
    {
        $em->remove($control);
        $em->flush();

        return $this->redirectToRoute('app_post_control');
    }
}
```