МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З. ҐЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **“ Проектування desktop-додатку на основі інтерфейсу програмування Windows Forms для супроводу рішень із виконання польових робіт ”**

Виконав: _____ст. гр. Кн-41_____

Спеціальності 122 – «Комп’ютерні науки»___
(шифр і назва)

_____Бойко Василь Васильович_____
(Прізвище та ініціали)

Керівник: ___к.т.н., доц. Луб П.М._____
(Прізвище та ініціали)

Рецензент: _____
(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

перший (бакалаврський) рівень вищої освіти
122 – «Комп'ютерні науки»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

_____Бойку Василю Васильовичу_____

1. Тема роботи: «Проектування desktop-додатку на основі інтерфейсу програмування Windows Forms для супроводу рішень із виконання польових робіт»

Керівник роботи: Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету від 123/к-с від 25.02.2025.

2. Строк подання студентом роботи – 16.06.2025.

3. Початкові дані до роботи: 1. Довідкова література. 2. Інформаційні джерела розробників мов програмування. 3. Особливості синтаксису мов програмування. 4. Стандарти роботи з Windows Forms. 5. Використання баз даних у десктопних додатках.

4. Зміст розрахунково-пояснювальної записки:

1. Аналіз іт-сервісів для агровиробництва;
2. Технології та етапи створення десктопного додатку;
3. Проектування десктопного додатку інструментами API Windows Forms;
4. Якість програмного забезпечення та тестування додатку;
5. Охорона праці та безпека в надзвичайних ситуаціях.
Висновки та пропозиції.
Бібліографічний список.
Додатки.

5. Перелік графічного матеріалу: 1 та 2 – Тема, актуальність, мета, завдання роботи; 3 – Аналіз ІТ-сервісів; 4 – Поняття десктопного додатку; 5 – Аналіз платформ та технологій; 6 – Проектування десктопного додатку; 7 – Проектування БД; 8 – Компоненти програми; 9 – Результати тестування додатку; 10 – Головні висновки роботи.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання – 25 лютого 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання	Приміт-ка
1.	Написання першого розділу та означення головних завдань роботи	25.02.2025 – 01.04.2025	
2.	Виконання другого розділу та формування головних показників для розрахунків	01.04.2025 – 01.05.2025	
3.	Виконання третього розділу та формування початкових даних	01.04.2025 – 01.05.2025	
4.	Виконання четвертого розділу та узагальнення отриманих результатів роботи	01.04.2025 – 01.05.2025	
5.	Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»	01.05.2025 – 01.06.2025	
6.	Завершення оформлення кваліфікаційної роботи. Перевірка подібності тексту.	01.06.2025 – 09.06.2025	

Студент _____ Бойку В.В.
(підпис)

Керівник роботи_____ Луб П.М.
(підпис)

УДК: 004.414.23:631.3.023.6

Кваліфікаційна робота: 63 с. текст. част., 37 рис., 2 табл., 12 слайдів, 21 джерел.

Проектування desktop-додатку на основі інтерфейсу програмування Windows Forms для супроводу рішень із виконання польових робіт. Бойку В.В. Кафедра ІТ. – Дубляни, Львівський НУВМБ, 2025.

Проаналізовано сучасний стан впровадження інформаційних технологій в аграрному секторі, зокрема тенденції в управлінні земельними ресурсами та плануванні агротехнічних робіт.

Розглянуто основні типи десктопних застосунків, проаналізовано особливості популярних операційних систем і фреймворків для їх розробки. З-поміж наявних технологій обґрунтовано вибір платформи Windows Forms як оптимального інструменту для реалізації проєкту, враховуючи вимоги до функціональності, простоти реалізації та офлайн-режиму роботи.

Розроблено концептуальну модель програмного забезпечення для агропланування, яка включає ключові сутності аграрної діяльності: поля, техніку, модулі, працівників і задачі. Створено додаток, що дозволяє планувати польові роботи з урахуванням доступності ресурсів та персоналу. Застосунок підтримує створення, редагування та збереження завдань, а також генерацію звітів у форматі PDF для подальшої передачі виконавцям.

Проведене тестування підтвердило відповідність додатку функціональним вимогам, його стабільну роботу та зручність інтерфейсу користувача.

Окрему увагу приділено вимогам охорони праці та безпеки в надзвичайних ситуаціях, що є невід’ємною частиною професійної діяльності у сфері ІТ.

Ключові слова: агропланувальник, десктоп-додаток, Windows Forms, планування робіт, програмне забезпечення.

Зміст

ВСТУП

Сучасне сільське господарство вже неможливо уявити без використання спеціалізованого програмного забезпечення. Такі цифрові інструменти дозволяють автоматизувати робочі процеси, аналізувати різні види даних, координувати дії працівників, управляти технікою й виконувати безліч інших завдань. Поєднання новітнього обладнання з програмними рішеннями забезпечує оптимізацію аграрних процесів, що дозволяє досягати високих результатів при мінімальних витратах ресурсів.

Сьогодні кожен фермер має можливість обрати відповідне програмне рішення, яке дозволить ефективно планувати та аналізувати всі етапи роботи — від підготовки ґрунту до збору врожаю. Застосування таких інструментів сприяє підвищенню продуктивності, зменшенню витрат та забезпечує точніше управління всіма аспектами аграрної діяльності, що є ключовим чинником стабільного розвитку господарства.

Головна мета програмного продукту — забезпечити просте й зручне планування аграрних робіт для малих фермерських господарств. Більшість сучасних агрододатків надто багатофункціональні, що ускладнює їх використання та вимагає тривалого навчання. Натомість проектований додаток буде орієнтований на вузьке коло завдань і максимально простий інтерфейс, що робить його ідеальним для новачків у агробізнесі та власників невеликих ділянок.

Завдяки інтуїтивно зрозумілому дизайну та фокусу на основних можливостях — таких як планування польових робіт і контроль за їх виконанням — наш додаток дозволить аграріям ефективно організовувати щоденну діяльність. Це спростить керування господарством і дасть змогу зосередитися на ключових аспектах: підвищенні врожайності, раціональному використанні ресурсів та забезпеченні довготривалого успіху і сталого розвитку.

Метою роботи є проектування desktop-додатку на основі API Windows Forms для планування та корегування перебігу сільськогосподарських польових

робіт.

Завдання роботи:

- проаналізувати сучасні додатки для планування робіт в агросекторі;
- розкрити особливості, вибрати інструментальні засоби для розробки десктопного додатку, який виконуватиме планування польових робіт у рільництві;
- описати концепцію побудови та проектування десктопного додатку на базі API Windows Forms;
- розробити додаток і провести його тестування.

РОЗДІЛ 1. АНАЛІЗ ІТ-СЕРВІСІВ ДЛЯ АГРОВИРОБНИЦТВА

1.1. Аналіз перспектив реалізації ІТ-проектів в агробізнесі України

У сучасному світі провідні країни активно впроваджують у сільське господарство концепцію «точного землеробства» — підхід, що передбачає детальне управління кожною ділянкою поля. Всі процеси — від підготовки ґрунту, посіву, внесення добрив до боротьби з бур'янами та шкідниками — автоматизуються, що дозволяє ефективно використовувати ресурси, зменшити витрати на насіння, добрива та засоби захисту рослин. Розвиток цифрових технологій в аграрному секторі має подібні засади до інших галузей: зменшення фінансових та часових витрат, точніше планування й розрахунки. Окрім цього, на ринку з'являються програмні продукти й технічні рішення, які сприяють оперативному впровадженню інновацій. Хоч аграрна сфера й приєдналася до цифрової трансформації дещо пізніше, вона впевнено надолужує відставання від фінансового, промислового та інших секторів економіки [5, 7]. То ж які ІТ-сервіси сьогодні пропонують в Україні?

AgriChain — це сучасна багатомодульна платформа, призначена для цифрової трансформації аграрного бізнесу. Вона об'єднує в собі інструменти для автоматизації бізнес-процесів і забезпечує комплексне управління всіма основними напрямками діяльності агропідприємства. До таких напрямків належать: управління земельним банком, виробничими процесами, моніторинг посівів, організація складської логістики, закупівлі та постачання матеріально-технічних ресурсів, контроль роботи техніки, облік ремонтів, а також логістика готової продукції.

Платформа дозволяє ефективно вирішувати широкий спектр завдань. Серед них: аудит та ведення земельного банку; моніторинг стану посівів; аналіз погодних умов, вологості ґрунту та його хімічного складу; сезонне планування і бюджетування; оперативне планування та облік робіт у полі; організація ремонтів техніки та транспорту; виконання складських операцій; автоматизоване

формування первинної документації; інтеграція з обліковими системами, зокрема 1С; а також створення бізнес-процесів і звітів за допомогою вбудованого конструктора.

Рисунок 1.1 – Система AgriChain та структура IT-рішень [2, 9]

Компанія Soft-Farm пропонує власне комплексне IT-рішення для сільськогосподарських підприємств. Його основна мета — інтегрувати дані з різних систем в єдиний формат, створивши прозору та надійну аналітичну платформу. Це забезпечує агровиробникам можливість приймати обґрунтовані управлінські рішення на основі достовірної інформації про всі аспекти господарської діяльності [2, 9].

Рисунок 1.2 – IT-інструменти Soft-Farm

До ключових напрямків розвитку цифрових технологій у сільському господарстві належать: супутникове спостереження за станом посівів, використання техніки для точного висіву й диференційованого внесення добрив, робота з великими обсягами даних (Big Data), впровадження автоматизованих систем управлінського обліку, застосування спеціалізованих датчиків, а також розвиток інформаційних технологій у агросфері (AgroIT).

Hummingbird Technologies — це компанія, яка спеціалізується на аналітичних рішеннях, побудованих на базі даних дистанційного зондування Землі. Обробка інформації виконується за допомогою штучного інтелекту та власних алгоритмів розпізнавання зображень. Завдяки таким технологіям клієнти компанії отримують можливість підвищити врожайність, оптимізувати використання добрив і засобів захисту рослин, а також приймати виважені агрономічні рішення вже на ранніх етапах виробничого циклу [2, 9].

Рисунок 1.3 – Дистанційне зондування в IT-послугах Hummingbird Technologies

Основні напрями діяльності Hummingbird Technologies включають: використання безпілотних літальних апаратів (БПЛА), супутниковий моніторинг, впровадження техніки точного висіву та диференційованого внесення, а також обробку великих масивів аграрних даних. Ці напрямки

дозволяють компанії надавати послуги з оцінки сходів, виявлення бур’янів, розробки рекомендацій щодо дозування регуляторів росту, азотних добрив і гербіцидів, картографування ризиків на полях, а також планування меліоративних заходів.

Компанія SAS також активно впроваджує інноваційні технології в аграрну сферу, сприяючи перетворенню традиційного агробізнесу на сучасне високотехнологічне підприємство [2, 9]. Серед її цифрових сервісів — моніторинг структури посівних площ, оперативне спостереження за полями, збирання детальної інформації про кожне поле, ведення електронного журналу агронома, прогнозування врожайності, управління земельним банком, контроль за переміщенням техніки, облік витрат пального, автоматичні сповіщення про відхилення в роботі, підключення додаткового обладнання та відображення статусу виконання аграрних робіт.

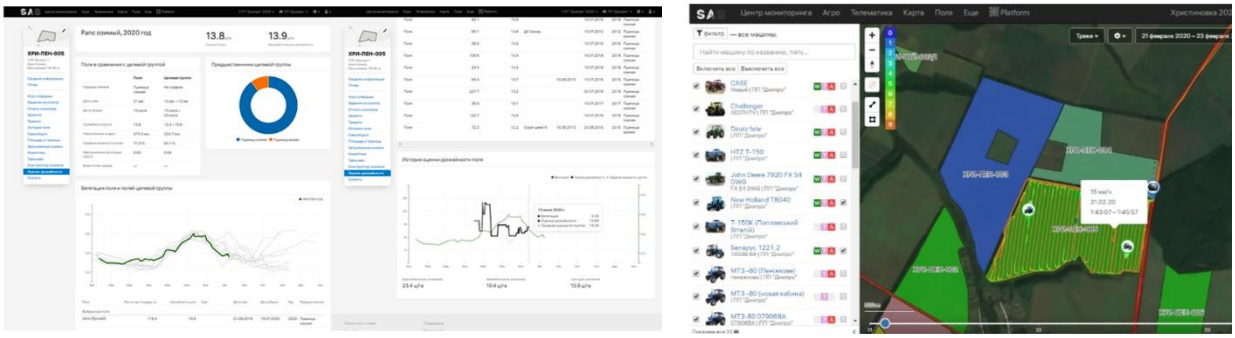

Рисунок 1.4 – SAS-технології для аграрних підприємств

DroneUA є одним із лідерів на українському ринку безпілотних рішень і виконує функції системного інтегратора технологій у галузі точного землеробства. Компанія має власні інженерні й виробничі структури, а також центр для обробки великих обсягів даних.

Окрім цього, DroneUA є офіційним імпортером і дистриб’ютором провідних брендів дронів на світовому ринку.

Рисунок 1.5 – Система IT-послуг DroneUA

До основних напрямів діяльності компанії належать: використання безпілотних апаратів у сільському господарстві, супутниковий моніторинг, аналіз великих масивів даних (Big Data) і розробка цифрових рішень у межах агроIT. Серед послуг, які надає компанія, — інтеграція безпілотних технологій та створення комплексних рішень для реалізації стратегії точного землеробства.

1.2 Аналіз додатків для планування робіт в агросекторі

Більшість агропланувальників є частиною комплексних програмних рішень, призначених для управління аграрним виробництвом. Знайти в мережі окремий програмний продукт, що був би орієнтований виключно на функцію планування в агросфері, майже неможливо. Тому під час дослідження ми звернули увагу на ті системи, які користуються популярністю серед фермерів. Основою для вибору слугувала статистика завантажень мобільних додатків у відповідних маркетплейсах, а також рейтинги на профільних платформах, орієнтованих на сільське господарство [10].

Першою системою, що ми розглянемо буде John Deere Operations Center. Це масштабна платформа для управління сільськогосподарськими даними, яка надає фермерським господарствам інструменти для ефективного планування, моніторингу та аналізу польових робіт. Платформа доступна у веб-версії, а також у мобільних додатках для Android та iOS, що забезпечує зручний доступ з будь-

якого пристрою в будь-який час і в будь-якому місці [10]. Доступ до функцій платформи є безкоштовним, і навіть не потрібно бути власником техніки даної марки.

Операційний центр складається з чотирьох основних функцій:

- Організація господарства. Система дозволяє налаштувати структуру підприємства — додати учасників команди, призначити техніку та визначити межі полів. Це спрощує внутрішні бізнес-процеси та забезпечує чітку координацію дій.
- Планування агротехнічних робіт. Завдяки інструментам для планування, користувачі можуть автоматизувати створення графіків виконання робіт. Це дає змогу уникнути затримок у проведенні агротехнічних заходів і покращити організацію праці.
- Контроль за польовими роботами. Платформа надає змогу у режимі реального часу відстежувати стан виконання робіт, продуктивність техніки та своєчасно реагувати на зміни або проблеми в полі.
- Обробка та аналіз даних. Аналітичні інструменти, інтегровані в систему, дозволяють глибоко аналізувати інформацію, отриману з полів, що сприяє прийняттю обґрунтованих і ефективних управлінських рішень.

Крім того, в Operations Center доступний зручний планувальник, за допомогою якого можна швидко створювати й редагувати плани польових робіт. Цю функцію можна використовувати як у веб-версії, так і через мобільний додаток, що забезпечує високу мобільність і гнучкість для користувачів.

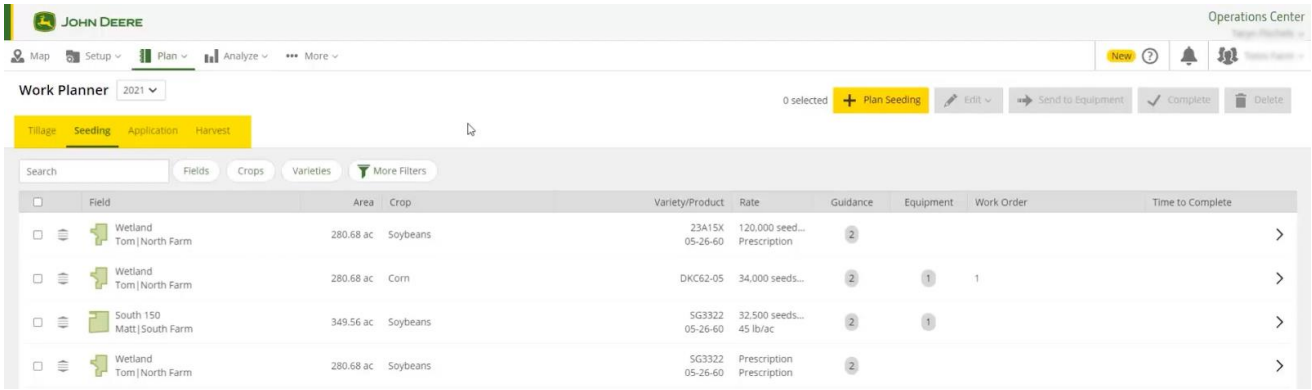

Рисунок 1.6 – Вікно формування завдань в платформі John Deere

Якщо обраною культурою для вирощування є соя, у John Deere Operations Center можна задати відповідні норми висіву та внесення добрив, орієнтовані саме на цю культуру. Система дозволяє встановити єдину цільову норму для всіх полів господарства, а за необхідності — скоригувати її для окремих ділянок.

Плани польових робіт і навігаційні маршрути, створені в системі, автоматично надсилаються на відповідне обладнання через бездротову технологію JD-Link. Це забезпечує миттєве відображення плану безпосередньо на дисплеї техніки в момент початку виконання завдання. До плану також можна додати додаткову інформацію — наприклад, номери нарядів або текстові інструкції, що допомагає працівникам точніше дотримуватись передбаченого сценарію виконання робіт.

Наступним прикладом є Trimble Ag Software — комплексне рішення для цифрового управління агровиробництвом, орієнтоване як на фермерів, так і на агрономів. Програмне забезпечення дає змогу оптимізувати управління полями, зменшити витрати та покращити врожайність [9]. Серед ключових переваг — широка інтеграція з іншими системами, зокрема Raven Slingshot, AGCO VarioDoc, AgCommand, John Deere Operations Center та ін.

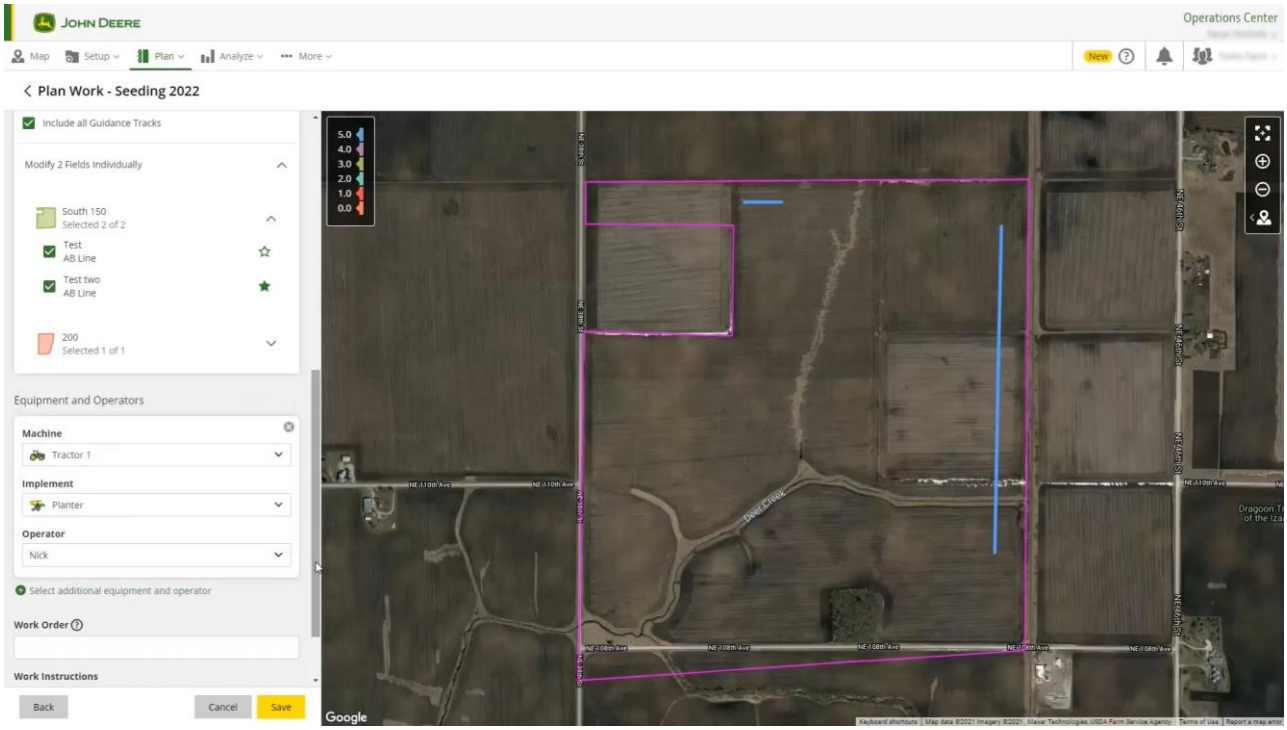

Рисунок 1.7 – Візуалізація планувальника задач

Основні функціональні можливості Trimble Ag Software включають:

- Картографію та ГІС — створення і керування картами полів із використанням геоінформаційних систем. Це охоплює моніторинг ґрунтів, аналіз урожайності й інші просторові дані.
- Планування аграрних робіт — інструменти для розробки графіків посіву, обробітку ґрунту, внесення добрив тощо.
- Моніторинг техніки — контроль роботи машин у реальному часі з доступом до даних про їхню продуктивність і технічний стан. Це дозволяє своєчасно виявляти проблеми та мінімізувати простої.
- Аналітику та звітність — функції для аналізу даних і створення звітів про врожайність, витрати, ефективність виконаних робіт тощо.

Ця система підходить як для невеликих фермерських господарств, так і для великих агропідприємств. Завдяки гнучкості та масштабованості, Trimble Ag Software може бути адаптована до конкретних потреб користувача, що забезпечує ефективне управління ресурсами та високу продуктивність.

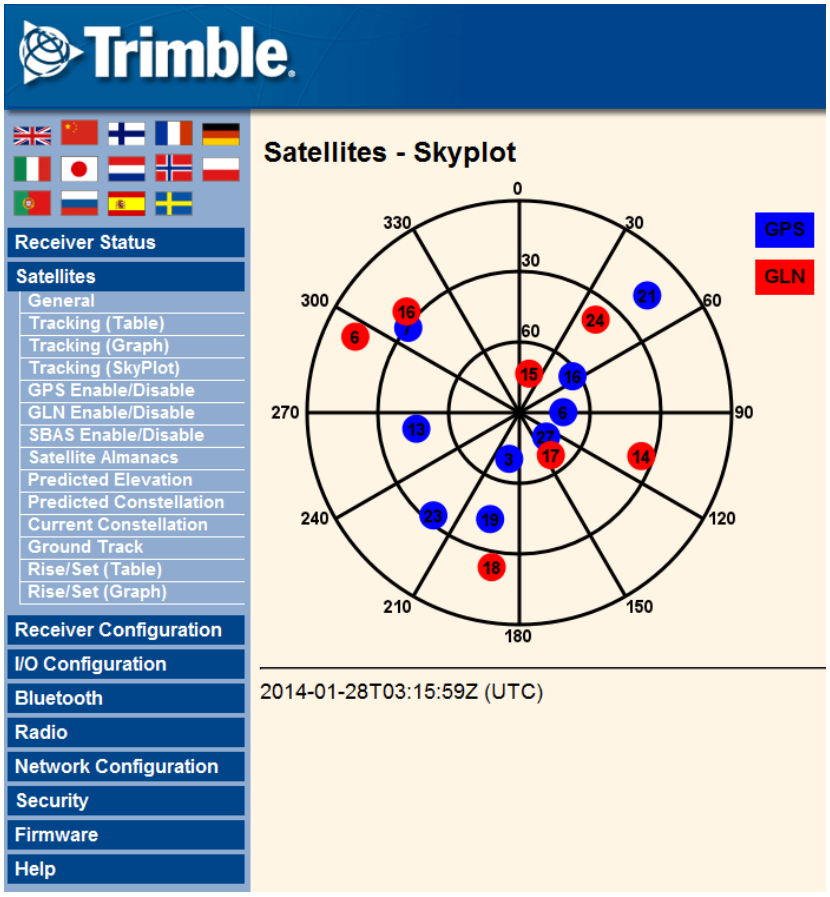

Рисунок 1.8 – Інтерфейс веб-версії Trimble

Ще одним важливим рішенням є Agrivi — універсальне програмне забезпечення для повного управління фермерськими господарствами, орієнтоване не лише на фермерів, а й на агропродовольчі компанії, кооперативи та фінансові установи [3]. Система допомагає оптимізувати фермерські процеси, підвищити ефективність і ухвалювати обґрунтовані управлінські рішення.

На відміну від конкурентів, таких як FarmLogs або Trimble Ag, Agrivi має розширений функціонал і орієнтований на повну цифровізацію господарської діяльності, не вимагаючи обов’язкового використання конкретного обладнання.

Система підтримує управління різними культурами й охоплює весь цикл аграрного виробництва — від планування до реалізації продукції. Для базового використання не потрібно підключати додаткове обладнання. Водночас Agrivi підтримує інтеграцію з такими ERP-рішеннями, як SAP чи Microsoft Dynamics, а також може працювати з датчиками, дронами та іншим обладнанням для підвищення точності обліку й моніторингу.

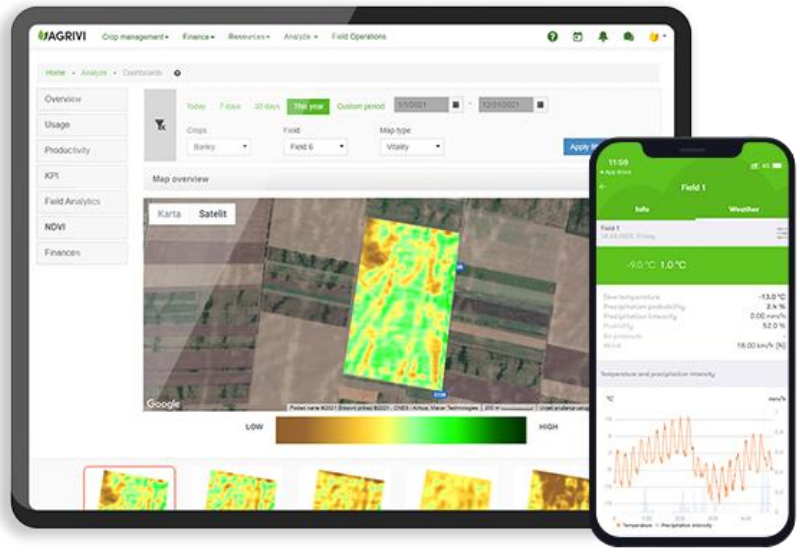

Рисунок 1.9 – Інтерфейс мобільного додатку Agrivi

Таким чином, John Deere Operations Center, Trimble Ag Software і Agrivi — це потужні цифрові інструменти, призначені переважно для середніх і великих господарств. Проте для невеликих фермерських підприємств ці рішення можуть виявитися надмірно складними або фінансово недосяжними. Крім того,

обмеження у вигляді відсутності десктопних версій також може бути важливим недоліком для окремої категорії користувачів.

Ці фактори підкреслюють необхідність створення менш громіздкого, але більш доступного і спеціалізованого програмного забезпечення для стаціонарних платформ, орієнтованого на потреби невеликих агровиробників.

1.3 Формулювання завдання створення додатку для планування агро-робіт

Великі комплексні цифрові системи, такі як John Deere Operations Center, Trimble Ag чи FarmLogs, дійсно здатні покрити більшість потреб сучасного аграрного бізнесу. Вони пропонують широкий функціонал, що охоплює всі етапи виробництва — від планування до аналітики. Однак варто зважати на те, що освоєння подібного програмного забезпечення потребує значного часу.

Також важливо враховувати масштаб господарства. Не завжди весь доступний функціонал подібних систем потрібен дрібному фермеру або власнику кількох ділянок, які використовуються для особистого вирощування культур. Наприклад, використання геоінформаційних систем, інтеграція з датчиками або підключення техніки з підтримкою цифрових модулів — усе це є актуальним переважно для великих агропідприємств. Для невеликого господарства подібні можливості можуть бути як надлишковими, так і недосяжними через відсутність відповідної технічної бази.

У ході аналізу сучасного ринку аграрного програмного забезпечення спостерігається чітка тенденція — домінування мобільних додатків і веб-платформ над десктопними аналогами. Тим не менш, це не означає, що програми, призначені для встановлення на персональні комп’ютери, втратили свою актуальність. Навпаки, саме для невеликих фермерських господарств десктопні автономні рішення можуть бути більш зручними, доступними та менш вимогливими до інтернет-з’єднання або наявності додаткових пристроїв.

Це дозволяє зробити висновок про перспективність розробки спеціалізованого автономного програмного забезпечення для персональних комп’ютерів, орієнтованого на малий і середній аграрний бізнес.

Підтвердити доцільність такого підходу можна, навівши декілька ключових аргументів на користь десктопних рішень:

- Продуктивність і швидкість. Програми для ПК зазвичай працюють швидше, адже напряму використовують ресурси комп’ютера і не залежать від якості інтернет-з'єднання.
- Безпека даних. Локальне зберігання інформації зменшує ризики витоку, що важливо для господарств, які не хочуть передавати свої дані через мережу.
- Робота офлайн. Десктопні програми працюють без постійного доступу до інтернету — це особливо актуально для сільської місцевості з нестабільним зв’язком.

У разі якісного проєктування архітектури, можна створити гнучкий і масштабований продукт. Спочатку — просте базове рішення для вузької цільової аудиторії, з можливістю додавання нових функцій у майбутньому. Такий підхід дозволяє уникнути витрат на хостинг, частих оновлень, характерних для вебплатформ, і гарантує повне володіння програмою з боку користувача.

Мета проєкту: створення десктопного додатку для планування аграрних робіт, адаптованого під потреби малих фермерських господарств.

Функціонал нашої програми буде зосереджений на ключових завданнях:

- Занесення даних про поля, техніку та працівників.
- Створення і редагування планів робіт.
- Відстеження виконання запланованих завдань.

Такий фокус дозволяє зробити продукт простим у використанні та ефективним у роботі. Розробка десктопного планувальника дасть змогу малим і середнім господарствам зручно керувати аграрними процесами без зайвих витрат та складних систем. Простота, доступність та надійність — основні принципи, на яких базуватиметься наш додаток.

РОЗДІЛ 2. ТЕХНОЛОГІЇ ТА ЕТАПИ СТВОРЕННЯ ДЕСКТОПНОГО ДОДАТКУ

2.1 Розгляд поняття десктоп-додатку

Десктопний додаток — це програмне забезпечення, яке запускається в середовищі операційної системи персонального комп’ютера. Його інсталяція відбувається за допомогою окремого інсталятора, після чого програма працює, використовуючи ресурси самого комп’ютера.

Ключовою перевагою такого типу програм є можливість функціонування без постійного доступу до інтернету. Хоча багато сучасних десктопних рішень підтримують роботу в мережі, це, здебільшого, потрібно для синхронізації даних між пристроями, спільної роботи з іншими користувачами чи отримання оновлень без потреби у повторному встановленні програми [19].

Десктопні та веб-додатки представляють два основні типи програмного забезпечення, які виконують різні завдання та мають свої унікальні характеристики. Головна відмінність між ними полягає в тому, що десктопні програми встановлюються безпосередньо на комп’ютер користувача та функціонують в межах операційної системи, зазвичай Windows. Натомість веб-додатки запускаються через браузер і працюють на основі серверної інфраструктури.

Як уже згадувалося, десктопні програми виконуються локально на пристрої, тоді як веб-додатки потребують постійного доступу до інтернету, адже їх запуск і обробка відбуваються на віддаленому сервері. Це є важливим критерієм при оцінюванні доцільності використання того чи іншого типу ПЗ.

Обидва підходи мають свої переваги й недоліки, і вибір між ними залежить від конкретних потреб користувача або бізнесу. Поки що немає універсального рішення, яке б повністю витіснило інші: десктопні, мобільні та веб-програми залишатимуться актуальними і надалі, принаймні до повної інтеграції користувачів у хмарні середовища через єдину інфраструктуру.

Для глибшого розуміння особливостей десктопних додатків варто порівняти їх із веб-додатками, наприклад, у вигляді таблиці (див. табл. 2.1).

Таблиця 2.1 – Відмінності десктоп- та веб-додатків

Десктоп - додатки	**Веб - додатки**
Запускаються та виконуються локально на комп’ютері	Виконуються на віддаленому сервері
Можуть працювати офлайн	Потребують доступу до інтернету
Можливість забезпечити кращу безпеку	Більше факторів ризику для кібер-атак
Швидкодія напряму залежить від апаратної частини комп’ютера користувача	Швидкість відповіді залежить від багатьох факторів
Необхідність встановлювати додаток	Додаток доступний без додаткового завантаження
-	Працює на усіх платформах, де доступний Браузер

Слід зауважити, що хоч багато сучасних десктопних рішень також потребують інтернет-з'єднання для певних функцій, більшість основних операцій вони виконують локально. Це дає змогу краще захищати дані — користувачі можуть самостійно встановити потрібне програмне забезпечення для безпеки. У випадку веб-додатків забезпечити повний контроль за всіма процесами значно складніше.

Також веб-додатки є більш вразливими до кіберзагроз, що може тимчасово або повністю обмежити доступ до сервісу. Проте їхня перевага полягає в універсальності доступу: для запуску не потрібно встановлювати додаткове ПЗ, достатньо будь-якого пристрою з браузером. Цей фактор і є однією з причин активного розвитку веб-платформ.

Крім того, варто зазначити, що існує кілька різновидів десктопного програмного забезпечення, які відрізняються за функціоналом та призначенням

[1715]:

- Автономні додатки — це класичні програми, що функціонують повністю незалежно та не потребують з’єднання з інтернетом для основної роботи.
- Клієнт-серверні додатки — програми, які встановлюються на комп’ютер, але для обміну або отримання даних підключаються до віддаленого сервера.
- Утиліти та доповнення — сюди відносяться інструменти, які покращують продуктивність операційної системи або розширюють можливості браузерів і програмного забезпечення.
- Системне ПЗ та сервіси — програми, що забезпечують функціонування ОС і дозволяють запускати інші додатки, належать до цієї категорії.
- Мультимедійні програми — до них належать засоби для відтворення відео, музики, подкастів та іншого медіаконтенту.

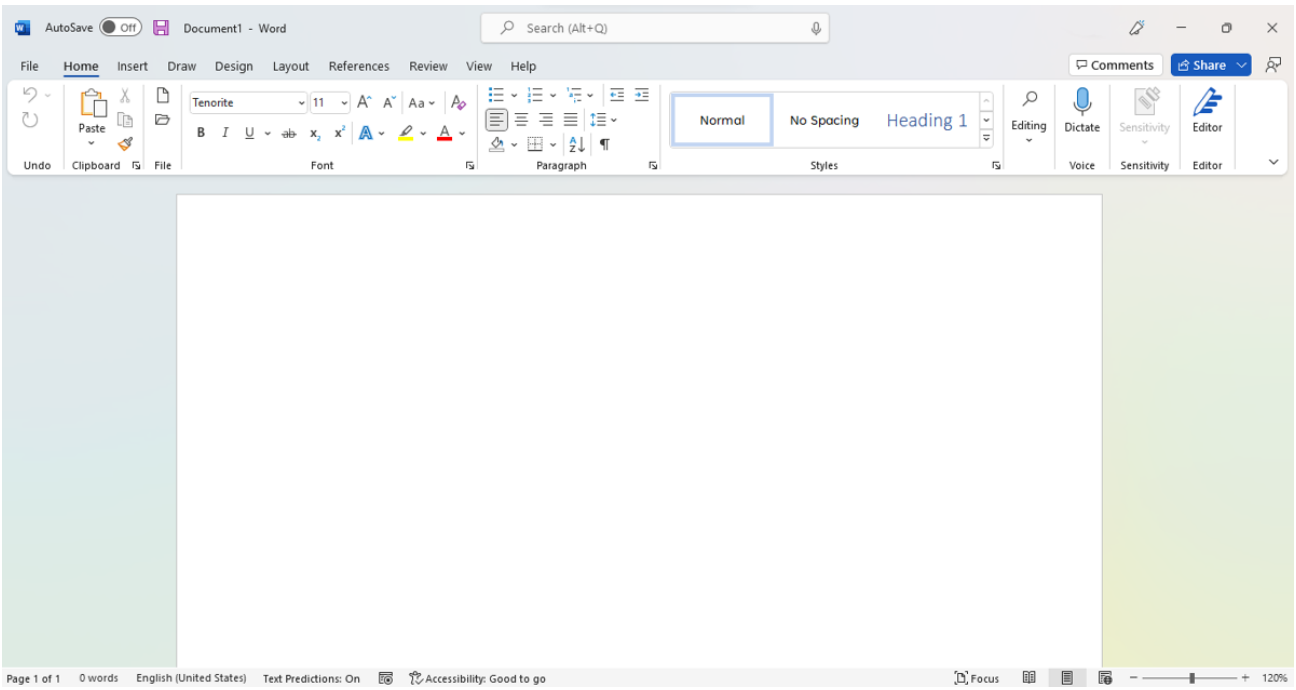

Рисунок 2.1 – Інтерфейс десктопної версії Microsoft Word

Десктопне програмне забезпечення працює під керуванням різних

операційних систем, зокрема Windows від Microsoft, macOS від Apple, а також відкритих систем на базі Linux.

Рисунок 2.2 – Логотипи популярних ОС

Згідно з даними Вікіпедії за 2024 рік, серед користувачів стаціонарних комп'ютерів та ноутбуків найпопулярнішою ОС залишається Microsoft Windows — її частка становить 72,22%. На другому місці — Apple macOS із показником 14,73%. Linux-дистрибутиви мають 3,88%, а Google ChromeOS — ще 2,45%. Оскільки ChromeOS базується на Linux, їх сумарна частка становить приблизно 6,33%. Невелика частка (0,01%) належить FreeBSD, а решта 6,7% припадає на менш поширені або спеціалізовані дистрибутиви Linux [20, 21].

2.2 Платформи та фреймворки для розробки десктопних додатків

Перш ніж перейти до аналізу інших операційних систем, зосередимось на платформі Windows, яка займає провідні позиції серед ОС за популярністю.

Операційна система Windows користується значним попитом у всьому світі завдяки своїй доступності, зручності та широкому спектру застосування. Вона має зрозумілий та інтуїтивний інтерфейс, який підходить як для новачків, так і для досвідчених користувачів. Windows активно застосовується як у комерційній сфері, так і для особистих цілей, зокрема в освіті, офісній роботі та розвагах.

Однією з її ключових переваг є наявність широкого вибору інструментів для

створення програмного забезпечення. Це дає змогу розробникам реалізовувати нативні додатки, тобто такі, що повністю адаптовані до функціоналу операційної системи та можуть ефективно використовувати її ресурси. Серед засобів розробки — Visual Studio, .NET, C#, C++, WinForms, WPF та інші технології, які забезпечують гнучкість у створенні додатків для різних сценаріїв використання.

Рисунок 2.3 – Інтерфейс Windows 11

Коли мова йде про створення нативного програмного забезпечення під Windows, першою згадується платформа .NET. Вона є відкритою технологією, яка дозволяє розробляти настільні, мобільні й вебдодатки, що можуть функціонувати на різних операційних системах. .NET включає в себе багатий набір інструментів, бібліотек і мов програмування, що забезпечують створення масштабованих, продуктивних і сучасних додатків [17, 18].

До ключових елементів екосистеми .NET можна віднести:

- Мови програмування: платформа підтримує широкий набір мов, серед яких C#, F#, Visual Basic, а також мови, сумісні з Common Language Infrastructure, такі як Eiffel, IronPython, PowerBuilder тощо.
- Модельні платформи: набір інструментів і бібліотек, що використовуються для розробки різних типів програм — настільних, мобільних

та вебдодатків.

- Середовище виконання .NET: компонент, що відповідає за запуск і виконання програмного коду.

Розробники використовують мови та платформи для створення програм, які потім виконуються за допомогою середовища .NET Runtime.

Сьогодні для розробки десктопних додатків у межах платформи .NET актуальними є три основні фреймворки:

WinForms — один із перших підходів до розробки настільних програм у Windows. Його інтерфейс будується шляхом перетягування елементів у візуальному редакторі, що значно полегшує процес створення простих інтерфейсів. Ідеально підходить для невеликих або середнього розміру проєктів, особливо на початкових етапах навчання.

WPF (Windows Presentation Foundation) — більш сучасна технологія, яка дозволяє створювати насичені, візуально привабливі інтерфейси з використанням XAML. Вона підтримує анімацію, тривимірну графіку, масштабування і гнучке зв’язування даних, що робить її добрим вибором для складніших проєктів з високими вимогами до UI.

.NET MAUI (Multi-platform App UI) — найновіше рішення в екосистемі .NET, яке дозволяє створювати кросплатформні додатки для Windows, macOS, Android і iOS, використовуючи єдину кодову базу. Ця технологія є логічним продовженням Xamarin.Forms і розширює можливості створення програм для кількох платформ одночасно, що робить її актуальною для розробників, які прагнуть охопити широку аудиторію.

Фреймворк Cocoa є основним нативним об'єктно-орієнтованим інтерфейсом прикладного програмування (API) для розробки додатків під macOS. Завдяки повній інтеграції з системними компонентами цієї операційної системи, додатки, створені з використанням Cocoa, демонструють високу продуктивність і забезпечують плавний користувацький досвід.

У порівнянні з SwiftUI, Cocoa разом із AppKit має кілька переваг. Зокрема, Cocoa — це перевірене часом рішення з розвиненою екосистемою бібліотек, що

дозволяє глибоко кастомізувати інтерфейс і легко інтегруватися з іншими частинами системи. Завдяки своїй зрілості та довготривалому розвитку, цей фреймворк часто показує кращу продуктивність у складних додатках. AppKit як складова Cocoa залишається стабільним інструментом, що забезпечує сумісність між різними версіями macOS.

Рисунок 2.4 – Інтерфейс MacOS Mojave

Наступною платформою для розгляду є Linux — вільна операційна система з відкритим вихідним кодом, яка славиться своєю адаптивністю та широкими можливостями налаштування. Основою всіх дистрибутивів Linux є його ядро, тоді як кожен дистрибутив — це власна збірка ПЗ та інтерфейсів, створена відповідно до певного призначення. Відомі дистрибутиви, такі як Ubuntu, Debian, Fedora та інші, орієнтовані на різні потреби — від користувачів-початківців до системних адміністраторів і розробників [14].

Одним із популярних інструментів для створення графічних інтерфейсів у Linux є GTK. Цей фреймворк надає великий набір UI-компонентів — від кнопок і полів вводу до діалогових вікон та вкладок. GTK вирізняється модульною структурою, можливістю гнучкого налаштування через CSS-подібні стилі, а

також хорошою інтеграцією з середовищем GNOME та іншими Linux-дистрибутивами [14].

Окремо слід відзначити Qt — кросплатформний фреймворк, що дозволяє створювати як графічні, так і консольні додатки для різноманітних операційних систем, включаючи Linux, Windows, macOS, а також мобільні платформи Android і iOS [12, 17]. Завдяки своїй гнучкості та потужному функціоналу, Qt є універсальним інструментом для професійної розробки.

Рисунок 2.5 – Інтерфейс робочого середовища GNOME

Після аналізу основних платформ для розробки настільного ПЗ, можна дійти висновку, що Windows є найбільш оптимальним вибором для реалізації нашого проекту. Створення додатків під macOS вимагає наявності комп’ютерів Apple, що значно підвищує вартість входу, а також потребує вивчення специфічних технологій. Linux, незважаючи на відкритість і широкі можливості, часто вимагає глибших технічних знань і не має такого великого обсягу навчальних ресурсів для створення графічних десктоп-додатків.

Крім того, цільова аудиторія нашого програмного забезпечення найімовірніше працює саме на Windows, тому вибір цієї ОС забезпечить більшу доступність, простоту підтримки та зменшення часу на адаптацію до платформи.

2.3 Опис технології створення Windows додатку

Розглянемо три найважливіші інструменти Microsoft для розробки настільних програм: Windows Forms, Windows Presentation Foundation та .NET MAUI. Кожен інструмент пропонує унікальні переваги та підходить для різних завдань, що дозволяє вам вибрати підхід, який найкраще відповідає вашим потребам у розробці сучасного, функціонального програмного забезпечення.

Першим інструментом у цьому огляді є Windows Forms. Windows Forms – одна з найстаріших технологій для створення графічних інтерфейсів на платформі .NET, яка широко використовується завдяки простоті використання та багатим навчальним ресурсам. Windows Forms надає розробникам зручний конструктор інтерфейсів у Visual Studio, який дозволяє їм створювати форми шляхом перетягування елементів.

Windows Forms використовує графічну бібліотеку Graphics Device Interface Plus (GDI+) для візуалізації інтерфейсу користувача. Цей API, розроблений Microsoft, дозволяє малювати прості графічні елементи, такі як кнопки, текстові поля, панелі та інші стандартні компоненти інтерфейсу користувача. GDI+ обробляє основні графічні завдання, такі як візуалізація форм і тексту, маніпулювання кольорами та зображеннями.

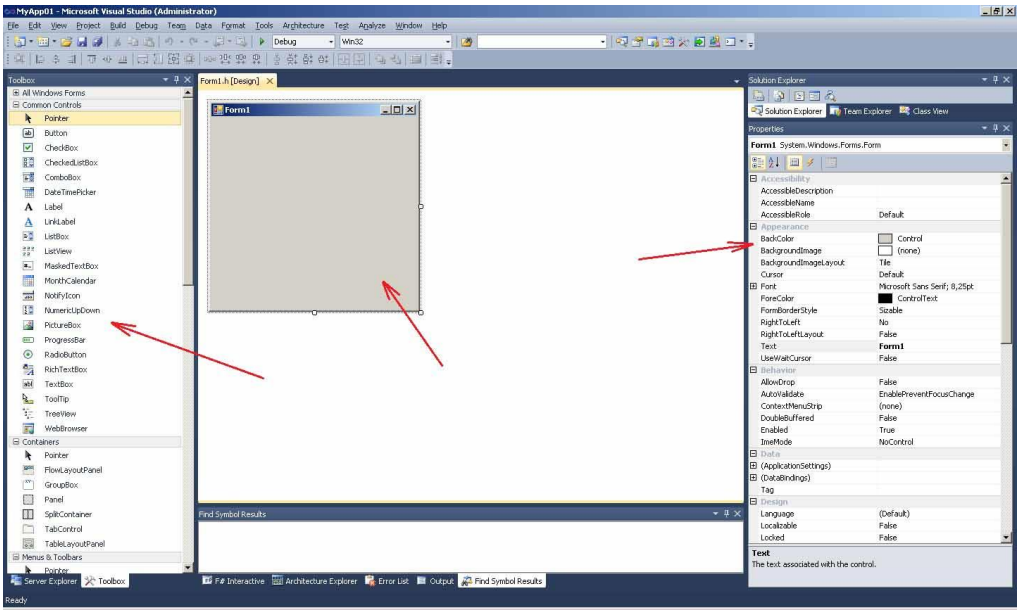

Рисунок 2.6 – Інтерфейс створення проектів Windows Forms в системі Microsoft Visual Studio

Кожен компонент (кнопка, текстове поле чи інший елемент) створюється як об'єкт і може бути легко маніпульований за допомогою коду або візуального дизайнера. Однак ця технологія має деякі обмеження, особливо щодо продуктивності та масштабованості. WinForms не підтримує складні графічні анімації та обмежена в адаптації до різних розмірів екрана.

Ще одним популярним інструментом є Windows Presentation Foundation (WPF). Ця передова технологія для розробки настільних програм у .NET дозволяє створювати інтерфейси зі складною графікою, анімацією та 3D-ефектами. На відміну від Windows Forms, WPF базується на XAML, мові розмітки, яка описує структуру та зовнішній вигляд інтерфейсу. Такий підхід забезпечує гнучкість для відокремлення логіки програми від інтерфейсу користувача та полегшує співпрацю між розробниками та дизайнерами.

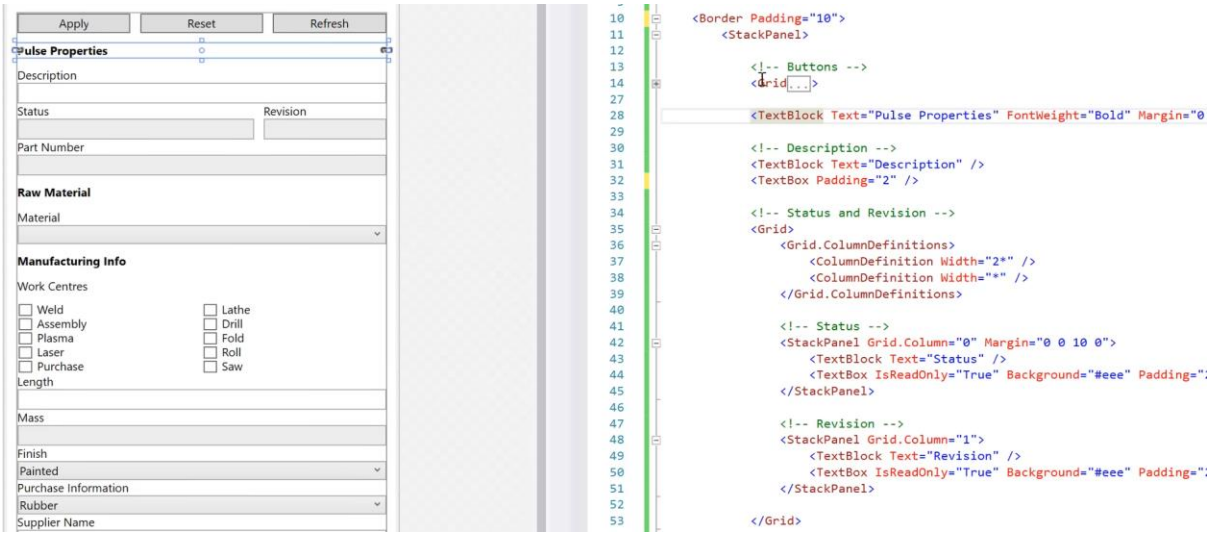

Рисунок 2.7 – Верстка інтерфейсу WPF додатка

На відміну від WinForms, WPF використовує апаратне прискорення та DirectX для візуалізації графіки, що дозволяє високопродуктивну реалізацію складних інтерфейсів. На практиці це означає, що WPF краще працює з мультимедійними елементами, великими обсягами даних та складною анімацією.

Ще однією важливою перевагою є адаптивність. Завдяки інтегрованій підтримці зв'язування елементів та адаптивного дизайну, WPF спрощує

масштабування користувацьких інтерфейсів до різних екранів.

Розроблений Microsoft, .NET MAUI (Multi-Platform App UI) – це уніфікований фреймворк для розробки кросплатформних додатків для Windows, Android, macOS та iOS. Розробники можуть написати код один раз та розгорнути його на кількох платформах, позбавляючи необхідності дублювати свою роботу в окремих нативних середовищах розробки. .NET MAUI успадковує основну функціональність Xamarin.Forms та розвиває її новими функціями та оптимізаціями.

. NET MAUI для Windows створює додатки на основі WinUI. Це означає, що користувацький інтерфейс додатка в середовищі Windows відповідає стандартам та зовнішньому вигляду додатка, розробленого за допомогою Windows SDK. Як результат, додатки .NET MAUI мають однаковий вигляд та функціональність у Windows, забезпечуючи користувачам однаковий досвід, подібний до нативних додатків[17].

Цей інструмент особливо корисний для додатків, розроблених для Windows, які працюватимуть на інших платформах. У таких випадках .NET MAUI – чудове рішення. Однак, якщо ви не плануєте використовувати його кросплатформно, це може бути не найкращим вибором. Ось кілька причин, чому:

- Нестандартний XAML: Синтаксис XAML .NET MAUI відрізняється від синтаксису WPF або Windows SDK, що вимагає додаткового часу на налаштування.
- Обмежені елементи керування: .NET MAUI наразі не підтримує Windows SDK, тому вам потрібно буде знайти сторонній елемент керування, щоб замінити його.
- Підвищена складність: Перенесення вашої програми на Windows вимагає додаткового перетворення, що може призвести до проблем зі стилем, API або макетом.

.NET MAUI — це нова, динамічна та перспективна технологія для створення сучасних кросплатформних програм. Її універсальні можливості надають розробникам широкий спектр можливостей для розробки та

розширення програм на кількох платформах. Технологія все ще перебуває в розробці, а ресурси обмежені, але очікується, що вона стане провідним рішенням у таких областях.

При порівнянні технологій Windows Forms і WPF, попри очевидні переваги WPF, у контексті нашого проєкту більш доцільним виглядає використання Windows Forms. Ця платформа — одна з класичних основ для створення десктопних додатків під Windows. Вона вирізняється простотою у використанні: розробнику достатньо перетягнути потрібні елементи на форму без необхідності глибоко занурюватися в нюанси їхнього налаштування. Це особливо важливо під час розробки невеликих або середньої складності програм, де на першому плані — швидкість створення функціонального інтерфейсу.

До того ж, Windows Forms дозволяє зосередити увагу саме на бізнес-логіці програми, не вимагаючи обов’язкового знання складних архітектурних підходів або додаткових технологій, які часто є частиною більш сучасних фреймворків. Враховуючи ці аспекти, можна з упевненістю стверджувати, що вибір цієї технології був обґрунтованим і доцільним.

У цьому розділі ми розглянули найпопулярніші операційні системи та відповідні технології для розробки настільного програмного забезпечення. Виходячи зі статистичних даних, Windows залишається домінуючою платформою як у сфері особистого користування, так і в бізнес-середовищі. З урахуванням цього та проаналізувавши доступні інструменти розробки для цієї ОС, ми зупинилися на Windows Forms як найкращому варіанті для створення прототипу агропланувальника. Ця технологія дозволяє реалізувати зручний та зрозумілий інтерфейс і водночас забезпечити достатню гнучкість і ефективність для вирішення поставлених завдань.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ДЕСКТОПНОГО ДОДАТКУ ІНСТРУМЕНТАМИ API WINDOWS FORMS

3.1 Базові інструменти Windows Forms

Як вже згадувалося, Windows Forms є складовою частиною платформи .NET і слугує інструментом для створення графічного інтерфейсу користувача (GUI) у настільних програмах. Ця бібліотека надає розробникам базовий функціонал для швидкої розробки віконних додатків із зручним, інтуїтивним інтерфейсом.

Основу технології складають стандартні елементи, що дають змогу формувати функціональні інтерфейси. Ключовими серед них є форми — структурні елементи, які виконують роль вікон програми, та контроли — інтерактивні елементи, через які здійснюється взаємодія користувача із додатком.

Форма виконує роль головного контейнера інтерфейсу та може мати різні призначення:

- Головна форма — основне вікно, яке відкривається під час запуску додатку.
- Модальна форма — діалогове вікно, що тимчасово блокує доступ до інших частин інтерфейсу, доки не буде закрите.
- Додаткові форми — допоміжні вікна, які відкриваються для виконання окремих задач і не заважають роботі з головною формою.

Інтерактивні елементи або контроли (наприклад, кнопки, поля введення, списки) забезпечують зв'язок користувача з програмою. Вони реагують на дії (події), як-от натискання або введення даних, та мають набір властивостей для налаштування зовнішнього вигляду й поведінки. За потреби розробники можуть створювати власні контролери, розширюючи функціонал стандартних елементів, використовуючи успадкування від таких базових класів, як Control, Button, TextBox тощо.

Для покращення зовнішнього вигляду інтерфейсу або додавання додаткових можливостей часто використовуються сторонні бібліотеки, такі як DevExpress чи Telerik, які надають розширені елементи керування та сучасні теми оформлення.

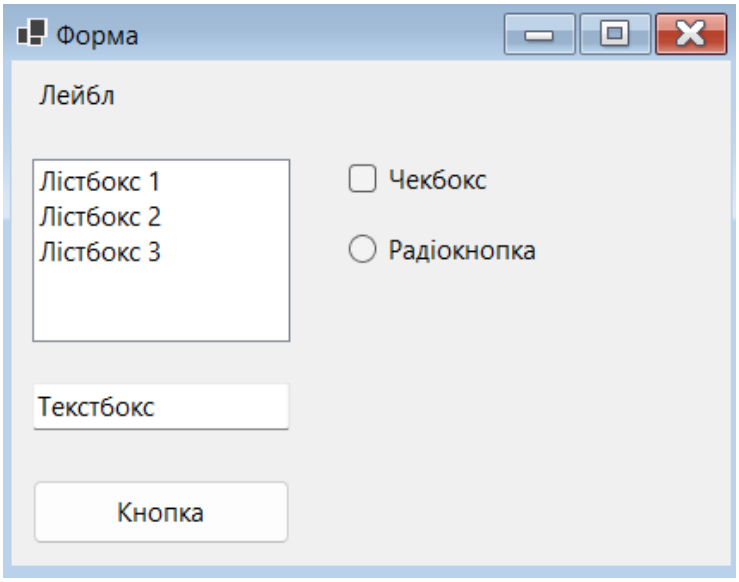

Рисунок 3.1 – Основні контроли форми

Однією з ключових складових під час проєктування графічного інтерфейсу є менеджери розташування (layout managers), які відповідають за правильне та логічне розміщення елементів на формі. Завдяки ним інтерфейс залишається впорядкованим і адаптивним до змін розмірів вікна чи інших умов.

До основних прикладів таких менеджерів належать:

- TableLayoutPanel — дозволяє компонувати елементи у вигляді сітки, що складається з рядків і стовпців, забезпечуючи структуроване розміщення контролів;
- FlowLayoutPanel — автоматично впорядковує елементи у вибраному напрямку (по горизонталі чи вертикалі), додаючи нові об'єкти послідовно один за одним.

Застосування таких інструментів значно спрощує процес створення адаптивного та зручного інтерфейсу, особливо у випадках, коли програма повинна коректно виглядати на різних екранах або під час зміни розмірів вікна.

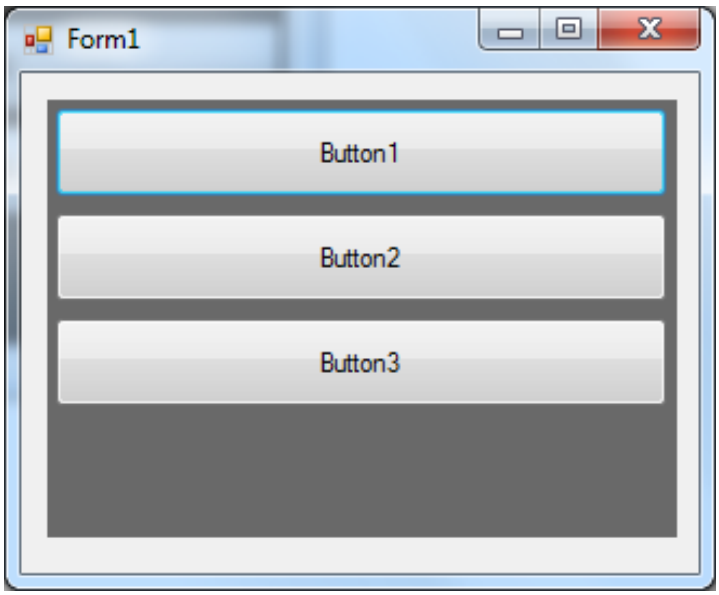

Рисунок 3.2 – Вирівнювання кнопок у функції "TableLayoutPanel"

Це дозволяє інтерфейсу залишатися зручним і структурованим навіть під час зміни розмірів вікна або масштабування його елементів.

Крім компоновки, не менш важливою складовою Windows Forms є механізм подій, який забезпечує реакцію програми на дії користувача або зміну її внутрішнього стану. Основні типи подій поділяються на:

- Інтерактивні події — відповідають за обробку взаємодії з користувачем: натискання кнопок, введення тексту, наведення курсора тощо.
- Системні події — сигналізують про зміну стану вікон, наприклад, відкриття чи закриття форми, зміна її розміру.
- Події введення з пристроїв — включають обробку натискань клавіш або рухів миші.

Обробка подій здійснюється через механізм підписки, при якому до певної події прив'язується метод, що виконується у відповідь на її виникнення. Наприклад, для елемента Button можна реалізувати подію Click, що запускає необхідну бізнес-логіку після натискання користувачем^

```
button1.Click += (sender, e) => MessageBox.Show("Кнопка натиснута!");
```

Рисунок 3.3 Фрагмент коду для події "Click"

Події в Windows Forms обробляються за допомогою механізму підписки, коли до конкретної дії прикріплюється відповідний метод-обробник. Наприклад, після натискання кнопки викликається метод, який призначений на подію Click.

Найпоширенішим інструментом для створення додатків із використанням Windows Forms є Visual Studio — інтегроване середовище розробки (IDE) від компанії Microsoft. Це середовище повністю підтримує роботу з проєктами на платформі .NET і дозволяє створювати, тестувати та відлагоджувати програми мовами C#, VB.NET тощо. Завдяки зручному інтерфейсу та широкому функціоналу Visual Studio є провідним вибором серед розробників для створення програм із графічним інтерфейсом у Windows [18].

Однією з головних переваг інтеграції Windows Forms із Visual Studio є наявність візуального редактора, який дозволяє будувати інтерфейси методом перетягування елементів із панелі інструментів (Toolbox) безпосередньо на форму. Цей підхід значно спрощує розробку UI, адже відповідний код створюється автоматично, що скорочує час на рутинні операції та дозволяє зосередитися на реалізації бізнес-логіки.

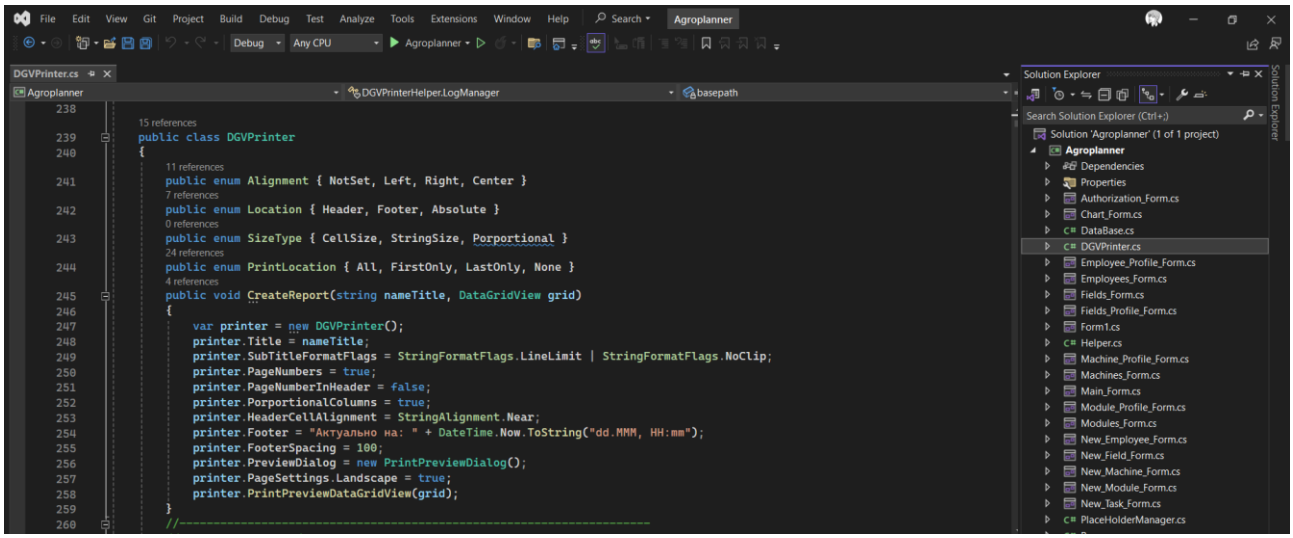

Рисунок 3.4 – Вікно редагування коду Visual Studio

Крім того, Visual Studio містить розширені засоби розробки: інтегровані функції налагодження, інструменти для аналізу продуктивності, підтримку систем контролю версій і можливість додавання сторонніх бібліотек. Це робить

процес розробки не лише ефективним, а й доступним навіть для розробників-початківців.

Платформа Windows Forms, у свою чергу, пропонує зручний та гнучкий набір засобів для створення десктопних програм. Поєднання простоти візуального дизайну та можливості тонкого налаштування дозволяє швидко реалізувати як базові, так і розширені функції, адаптовані під конкретні потреби користувачів. Саме тому Windows Forms залишається одним із найоптимальніших рішень для оперативної розробки стабільних та функціональних програм для операційної системи Windows.

3.2 Проектування десктопного додатку

Перш ніж формулювати базові концепції додатку, необхідно врахувати, що розроблюване програмне забезпечення належить до категорії повністю офлайн-додатків. Це означає, що питання обміну даними між системами потребує попереднього аналізу — зокрема, варто з’ясувати, чи взагалі такий обмін буде необхідним у рамках функціонування програми.

Аналізуючи сучасні програмні рішення, що підтримують аграрний бізнес у процесах планування, можна помітити спільну рису: всі вони оперують поняттям польової задачі, яка є базовою одиницею планування. Така задача, як правило, включає кілька обов’язкових елементів:

- Поле, на якому здійснюються роботи;
- Вид виконуваних робіт;
- Сільськогосподарська техніка;
- Додаткове навісне обладнання або модулі;
- Період виконання (дата початку та завершення);
- Призначений оператор.

Хоча в деяких системах враховуються додаткові параметри, наведені вище компоненти становлять ядро будь-якої планувальної структури [13].

Типи робіт, які виконуються на полі, можна умовно згрупувати за функціональним призначенням. Однією з найбільш ресурсомістких категорій є обробка ґрунту, яка охоплює такі операції, як оранка, боронування, культивація, дискування, глибоке рихлення і мульчування. Метою цих процедур є покращення структури ґрунту, знищення бур'янів та оптимальна підготовка площ до сівби.

Інші ключові типи робіт включають:

1. Посівні операції;
2. Внесення добрив;
3. Полив;
4. Захист рослин (обробка пестицидами);
5. Збирання врожаю.

Кожна з цих груп має критичне значення для досягнення стабільних агровиробничих результатів.

При плануванні задач важливо розуміти специфіку сільськогосподарської техніки, яку можна умовно поділити на трактори та комбайни. Комбайни спеціалізуються на збиранні врожаю і часто адаптовані під певні культури — пшеницю, кукурудзу, соняшник тощо. Вони поєднують у собі механізми зрізання, обмолоту і очищення, забезпечуючи високу швидкість і ефективність збору.

Трактори є багатофункціональною технікою, яку можна використовувати майже на всіх інших етапах агровиробництва — від підготовки поля до внесення добрив. Це досягається шляхом підключення відповідних модулів або навісного обладнання: плугів, сівалок, обприскувачів, розкидачів тощо. Завдяки цьому трактори демонструють високу гнучкість у виконанні різноманітних задач.

Ще одним критичним елементом є оператор техніки, який повинен мати відповідну кваліфікацію та дозвіл для керування певним типом машин. У контексті планування це дозволяє уникнути конфліктів у розкладі, рівномірно розподіляючи робоче навантаження та забезпечуючи раціональне використання людських ресурсів. Також важливо враховувати зайнятість оператора в інших

задачах, щоб мінімізувати простої і забезпечити ефективну реалізацію всього робочого плану.

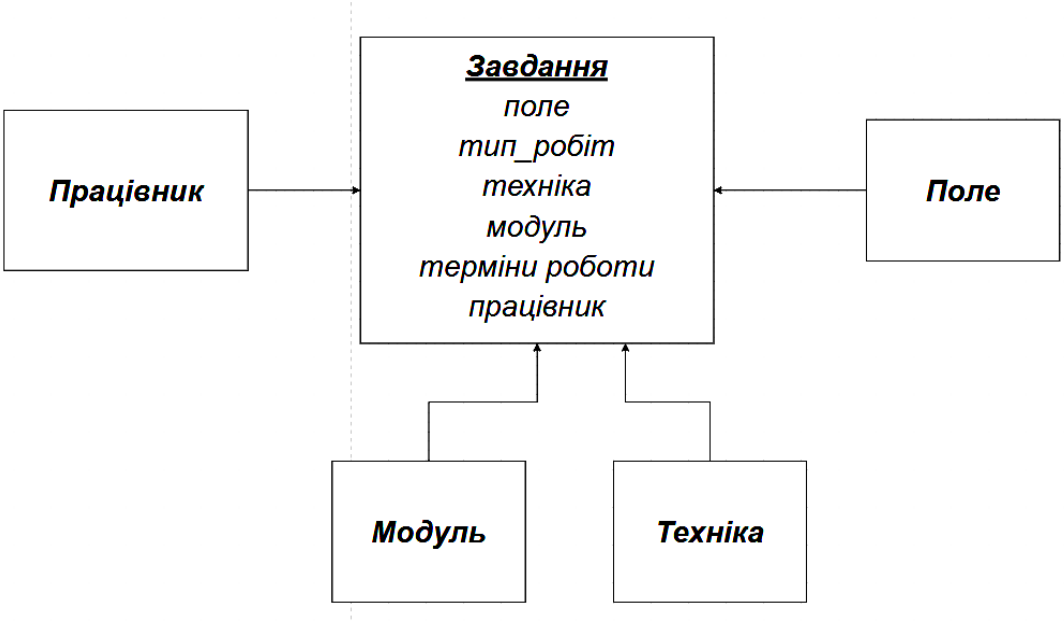

Рисунок 3.5 – Ключові елементи додатку

Те саме стосується техніки та модулів: важливо переконатися, що необхідне обладнання не буде зайняте в інших завданнях під час планування нової роботи. Це дозволяє забезпечити безперебійний робочий процес і оптимізувати використання технічних ресурсів.

Також потрібно дати користувачу можливість вносити нові записи усіх складових цього процесу, редагувати та видаляти їх. Але ключовим моментом буде об’єднання цих сутностей у так звані завдання відповідно до зайнятості цих об’єктів в інших завданнях.

Після того як ми сформували ключову концепцію нашого планувальника варто продумати сценарій використання даного програмного забезпечення.

Виходячи з усіх накладених факторів утворюється напевно, що єдиний сценарій використання цього ПЗ. Користувач, а саме власник бізнесу чи відповідальна особа встановлює дистрибутив на свій ПК, створює усі необхідні поля і одразу може планувати свої роботи. Однак тут з'являється одна складність. Наша програма є повністю офлайн додатком, тому немає можливості одразу в режимі онлайн відправляти інструкції з роботою виконавцям.

Відштовхуючись від вищесказаного, було прийнято рішення створити список завдань у вигляді файлу, який можна пересилати будь-яким зручним способом виконавцю. В цьому випадку найуніверсальнішим варіантом, було створення PDF файлу з таблицею робіт. Як правило більшість операційних систем можуть за замовчуванням відкривати цей формат.

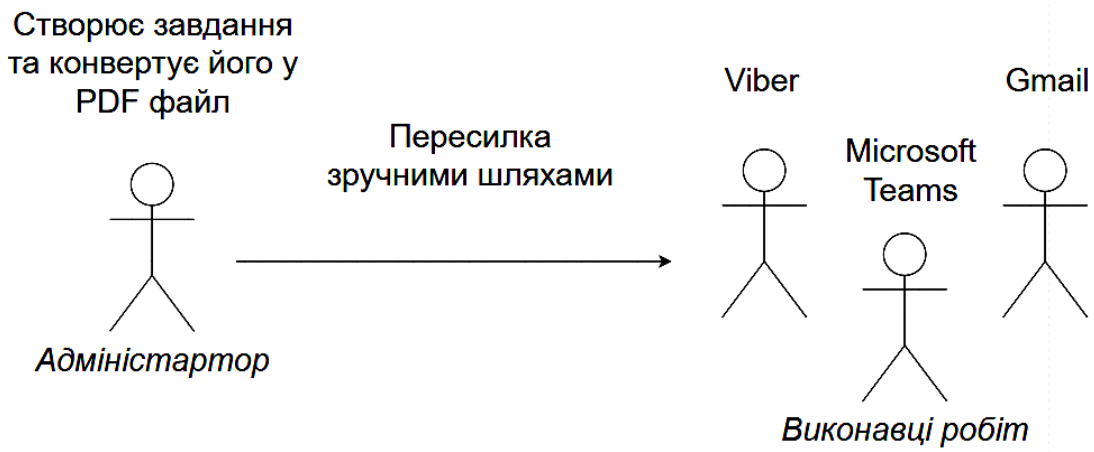

Рисунок 3.6 – Сценарій поширення файлів

Якісний UX та UI дизайн відіграють ключову роль у забезпеченні зручності та інтуїтивності користування програмою. Після того як основна концепція та сценарій використання додатку сформовані, можна розглянути графічне представлення програми.

Перед нами стоїть завдання у відображенні всіх ключових аспектів нашого планувальника та елементів керування для їх функцій. Нижче запропонований вигляд вікна нашої програми у вкладці "Працівники".

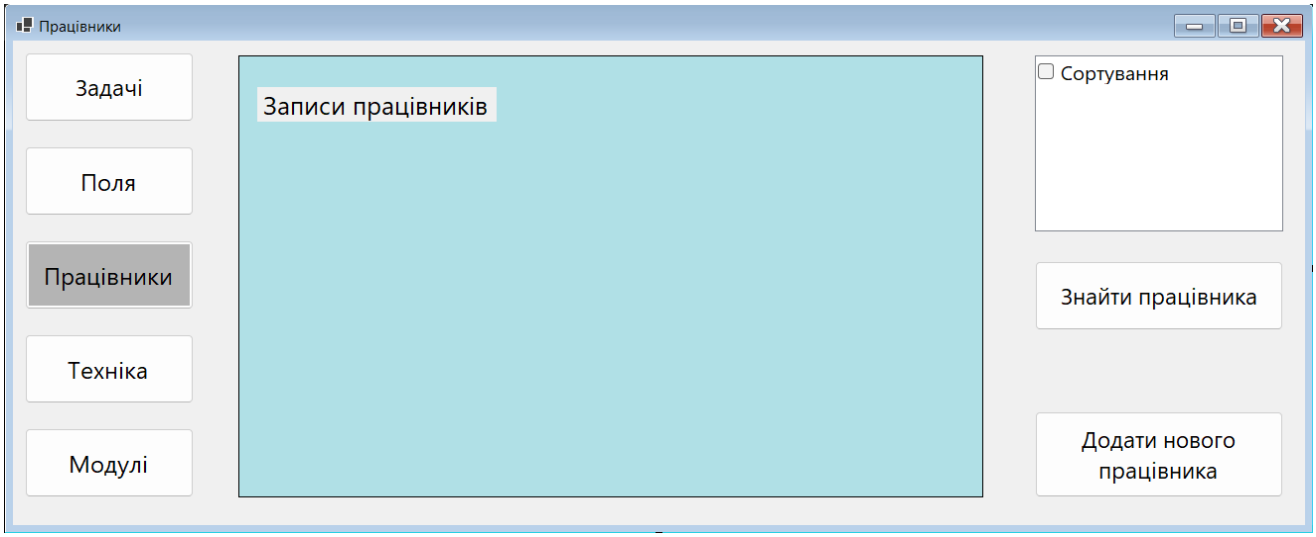

Рисунок 3.7 – Прототип вкладки "Працівники"

Як бачимо зліва розташована панель переключення між вікнами у вигляді вкладок. Дана візуальна композиція буде притаманна всім іншими вкладкам з нашими сутностями.

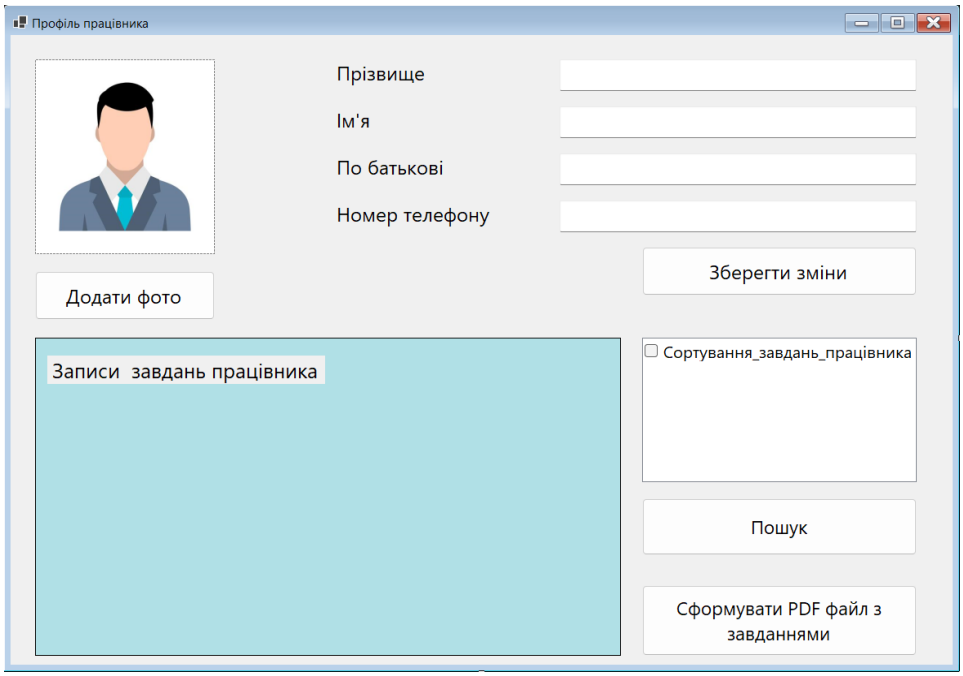

Рисунок 3.8 – Прототип профільної сторінки окремого працівника

Також ми додали можливість сортувати та створювати нові записи у цих вкладках.

Щоб працювати з окремим записом певної сутності створимо приблизний вигляд вікна профіля працівника.

У цьому вікні можна буде детальніше розглянути інформацію про конкретний запис та внести зміни. Також буде можливість переглянути усі завдання працівника, відсортувати їх та створити PDF файл на їх основі.

3.3 Розробка ключових блоків агропланувальника

Після проведення попереднього аналізу ніші, затвердження концепції та створення дизайн-макета можна переходити до етапу розробки. Було вирішено почати з проектування БД, оскільки основною функцією нашої програми є

робота з записами. Правильно спроектована база даних забезпечить ефективне зберігання та швидкий доступ до необхідної інформації.

При створенні нових таблиць ми одразу враховували, яка інформація в них зберігатиметься та які це будуть типи даних. Це дозволило краще зрозуміти, як працювати з сутностями бази даних, а також уникнути труднощів із її розширенням і модифікацією в майбутньому.

Nazarii_Leskiv\SQL...B1 - dbo.employees

Column Name	Data Type	Allow Nulls
id_employee	int	☐
last_name	nvarchar(50)	☑
first_name	nvarchar(50)	☑
patronymic	nvarchar(50)	☑
phone_number	nvarchar(50)	☑
place_of_residence	nvarchar(MAX)	☑
job_title	int	☑
driving_license_category	int	☑
is_still_working	int	☑
notes	nvarchar(MAX)	☑
image	varbinary(MAX)	☑

Рисунок 3.9 – Таблиця "employee"

Для прикладу, у таблиці "employee" ми зазначили всі необхідні параметри та відповідні їм типи даних. Особливо варто виділити колонки "job_title" та "driving_license_category". У нашій програмі реалізовано наступну логіку: кожен працівник може мати різні категорії на сільгосптехніку. Якщо працівник має категорію лише на трактор, то він не може працювати комбайнером, і навпаки. Крім того, є універсальні працівники, які мають обидві категорії. Окремо ми додали колонку "is_still_working", щоб знати, чи працівник все ще працевлаштований для уникнення можливих помилок.

Додатково були створені ще чотири таблиці: "machines", "modules", "fields" та "tasks". Останню таблицю розглянемо детальніше.

Таблиця "tasks" була спроектована таким чином, щоб включати ідентифікатори всіх інших складових об'єктів.

Вона також містить часові рамки для виконання завдання та статус його виконання. Важливою є колонка "notes", яка містить нотатки.

Nazarii_Leskiv\SQL...nnerDB1 - dbo.tasks

	Column Name	Data Type	Allow Nulls
🔑	id_task	int	☐
	id_field	int	☐
	type_of_work	int	☑
	time_to_start	datetime	☑
	time_to_finish	datetime	☑
	id_employee	int	☐
	id_machine	int	☐
	id_module	int	☑
	status	int	☑
	notes	nvarchar(MAX)	☑
	image	varbinary(MAX)	☑

Рисунок 3.10 – Елементи таблиці "tasks"

Це особливо корисно, коли, наприклад, виконується робота з внесення добрив — можна відразу зазначити, скільки добрив використано, які саме добрива застосовані тощо. Також важливим є атрибут "type_of_work", що відповідає за тип виконуваних робіт і слугує своєрідним фільтром при виборі техніки та модулів, що підходять для цього типу робіт.

Наступним кроком було встановлення зв’язків між таблицями для забезпечення взаємозв'язку даних і уникнення можливих помилок. Це дозволило інтегрувати різні компоненти системи, забезпечуючи зручність роботи з даними та їхню узгодженість. Грамотно спроектовані зв’язки підвищують ефективність використання та аналізу інформації, що є важливим для стабільного та успішного функціонування програмного продукту.

На рисунку 3.11 наведено зв'язки типу "один до багатьох" з таблицею "tasks", де, наприклад, один працівник може бути задіяний у багатьох завданнях. Зовнішні ключі в таблиці "tasks" посилаються на первинні ключі інших таблиць, забезпечуючи зв'язок між даними за їх унікальними ідентифікаторами.

Завершивши проектування бази даних, ми перейшли до безпосередньої розробки. Спочатку створили новий проект Windows Forms з назвою "AgroPlanner". Наступним кроком встановили необхідні компоненти для взаємодії з SQL сервером та пакетом візуалізації діаграм у Windows Forms, які знадоблятьс я нам для відображення гістограм.

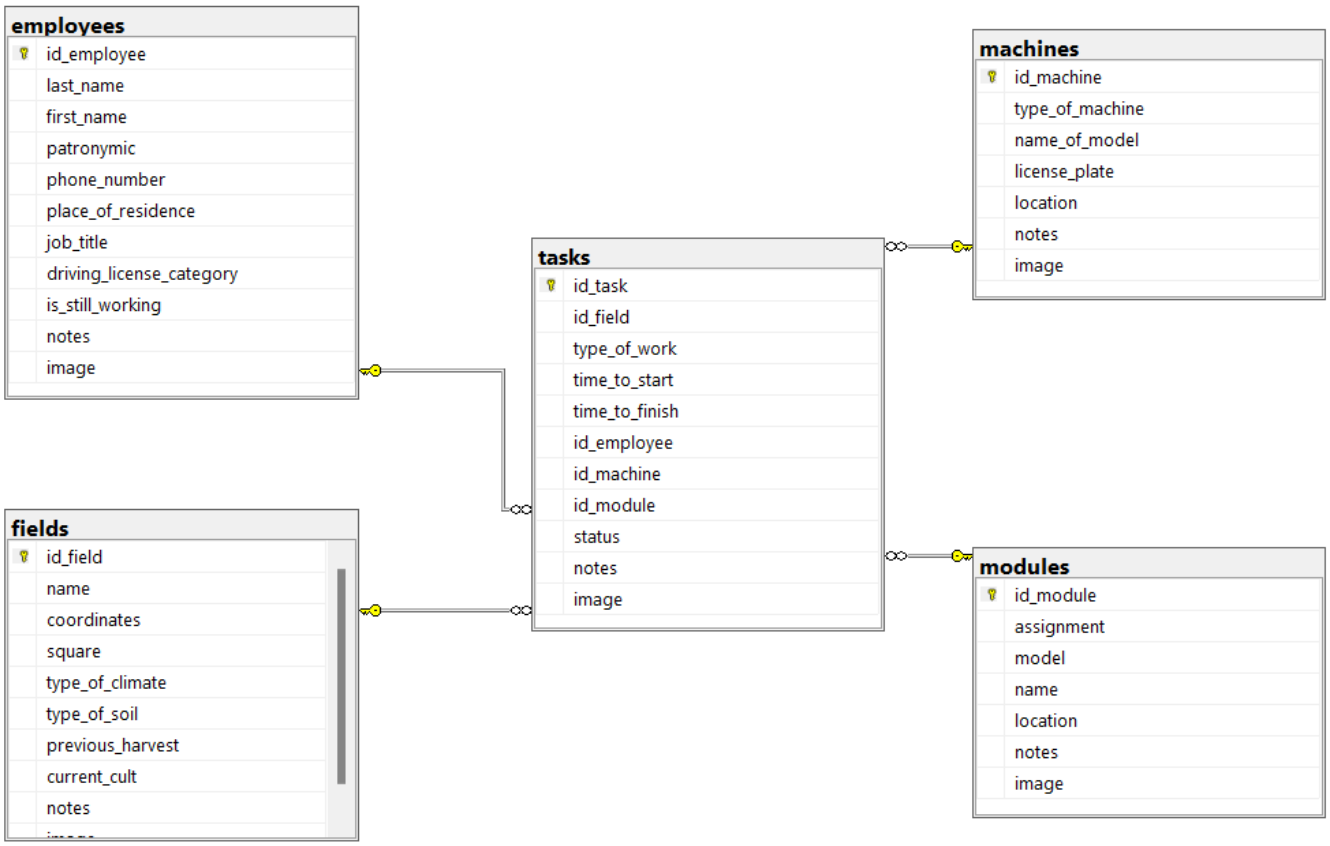

Рисунок 3.11 – Зв’язки між таблицями БД

Після цього ми створили клас "DataBase" для взаємодії з базою даних. Він буде відповідати за відкриття, закриття та управління з'єднанням з БД.

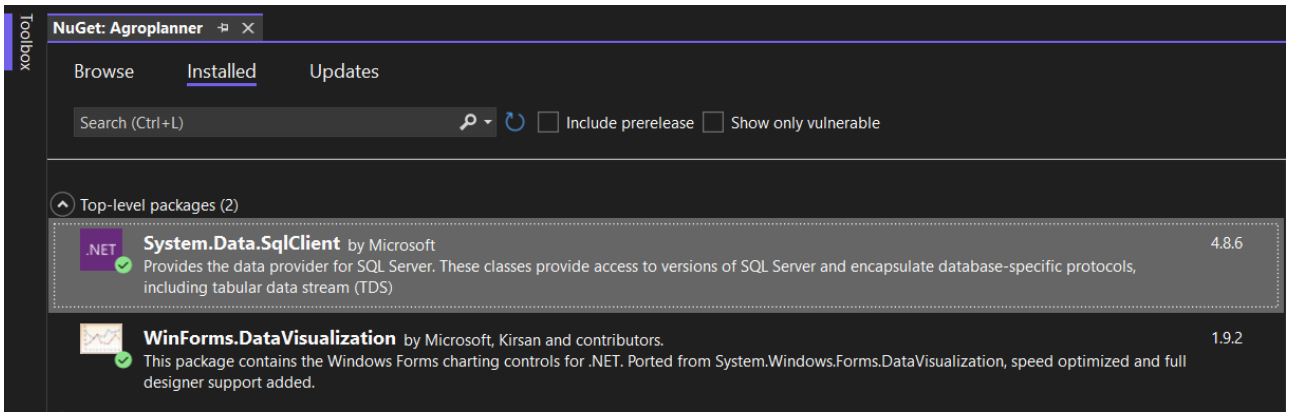

Рисунок 3.12 – Встановлені компоненти

Під час розробки цього класу було приділено особливу увагу безпеці даних та ефективності взаємодії з базою даних, щоб забезпечити надійність і високу швидкодію цього фрагменту коду.

```
46 references
internal class DataBase
{
    SqlConnection sqlConnection = new SqlConnection(@"Data Source = NAZARII_LESKIV\SQLEXPRESS;
    Initial Catalog = AgroplannerDB1; Integrated Security=True");

    40 references
    public void openConnection()
    {   if (sqlConnection.State == System.Data.ConnectionState.Closed)
        {
            sqlConnection.Open();
    }   }

    39 references
    public void closeConnection()
    {   if(sqlConnection.State == System.Data.ConnectionState.Open)
        {
            sqlConnection.Close();
    }   }

    41 references
    public SqlConnection GetConnection() { return sqlConnection; }
}
```

Рисунок 3.13 – Клас "DataBase" взаємодії з базою даних

Після успішної перевірки на з’єднання з БД, ми перейшли до створення наших форм згідно попередньо створеного шаблону. Було додано усі необхідні компоненти, а саме: кнопки, текстові поля, випадаючі списки, надписи.

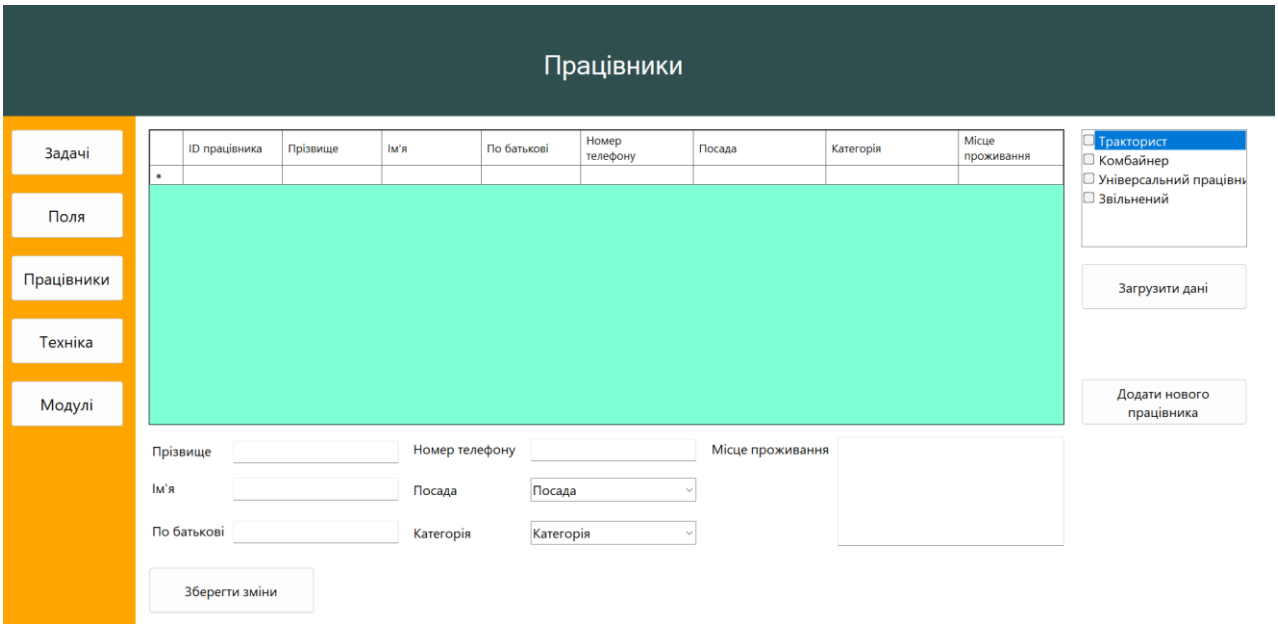

Рисунок 3.14 – Форма "Працівники"

Головним елементом наших вкладок є компонент DataGridView, який використовується для візуального відображення даних у вигляді таблиці. DataGridView забезпечує зручне представлення записів усіх сутностей програми,

дозволяючи користувачам легко переглядати та взаємодіяти з даними. Цей компонент дуже популярний у розробці десктопних додатків, оскільки підтримує сортування, фільтрацію, редагування та інші функції, що спрощують роботу з великими обсягами інформації.

В першу чергу ми створили блок коду, що відповідає за заповнення таблиці даними.

```
1 reference
private void ReadSingleRow(DataGridView dgv, IDataRecord record)
{
    dgv.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetString(4), JobTitleCheck(record.GetInt32(6)),
        CategoryCheck(record.GetInt32(7)), record.GetString(5), record.GetInt32(6), record.GetInt32(7));
}

1 reference
private string GetQuery()
{
    string query = $"SELECT id_employee, last_name, first_name, patronymic, phone_number, place_of_residence,  job_title, driving_license_category " +
        $"FROM employees WHERE is_still_working = {UnemploymentItemCheck()}";

    for (int i = 0; i < (checkedListBox_Sorting.Items.Count - 1); i++)
    {
        if (checkedListBox_Sorting.GetItemChecked(i))
        {
            query += $" AND job_title = {i} ";
        }
    }
    return query;
}
```

Рисунок 3.15 – Методи "ReadSingleRow" Та "GetQuery"

Метод "ReadSingleRow" додає нові записи до таблиці, використовуючи інтерфейс "IDataRecord", який забезпечує доступ до одного рядка даних, отриманого з бази даних. У тілі методу ми витягуємо значення з відповідних колонок запису та додаємо їх до таблиці.

Метод "GetQuery" повертає сформований запит, враховуючи, чи користувач застосував певне сортування працівників, що дозволяє отримати відповідні дані з бази.

```
3 references
public void RefreshDataGridView(DataGridView dgv) // Оновлює DataGridView
{
    dgv.Rows.Clear(); // Очищаєм рядки після кожного оновлення
    db.openConnection();
    SqlCommand cmd = new SqlCommand(GetQuery(), db.GetConnection());
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        ReadSingleRow(dgv, reader);
    }
    reader.Close();
    db.closeConnection();
}
```

Рисунок 3.16 – Метод "RefreshDataGridView"

Метод "RefreshDataGridView" очищує всі рядки у таблиці, відкриває з'єднання з базою даних, виконує SQL-запит отриманий з методу "GetQuery", і читає результати за допомогою функції "SqlDataReader". Для кожного запису, отриманого з бази даних, викликається метод "ReadSingleRow", який додає рядок до DataGridView. Після того як записи у таблиці закінчуються метод "SqlDataReader" припиняє свою роботу, а з'єднання з базою даних закривається.

Після додавання кількох записів у таблицю та натискання кнопки "Завантажити дані" можемо спостерігати коректну роботу програми з відображенням даних.

	ID працівника	Прізвище	Ім'я	По батькові	Номер телефону	Посада	Категорія	Місце проживання
▸	3	Сидоренко	Михайло	Петрович	0682345678	Комбайнер	Категорія на комбайн	Львівська обл., с...
	4	Коваль	Василь	Іванович	0693456789	Тракторист	Категорія на трактор	Львівська обл., с...
	6	Шевченко	Дмитро	Андрійович	0675678901	Універсальний праців...	Обидві категорії	Львівська обл., с...
	7	Мельник	Олексій	Олександрович	0686789012	Тракторист	Категорія на трактор	Львівська обл., с...
	9	Романенко	Юрій	Михайлович	0668901234	Універсальний праців...	Обидві категорії	Львівська обл., с...
	10	Лисенко	Артем	Петрович	0679012345	Тракторист	Категорія на трактор	Львівська обл., с...
	12	Ткаченко	Богдан	Сергійович	0691234567	Універсальний праців...	Обидві категорії	Львівська обл., с...
	13	Бондаренко	Іван	Андрійович	0662345678	Тракторист	Категорія на трактор	Львівська обл., с...
	15	Мартинюк	Євген	Юрійович	0684567890	Універсальний праців...	Обидві категорії	Львівська обл., с...
	16	Федоренко	Максим	Петрович	0695678901	Тракторист	Категорія на трактор	Львівська обл., с...
	18	Захаренко	Ростислав	Ігорович	0677890123	Універсальний праців...	Обидві категорії	Львівська обл., с...
	19	Кузьменко	Сергій	Миколайович	0688901234	Тракторист	Категорія на трактор	Львівська обл., с...
*								

Рисунок 3.17 – Структура записів "employee" у DataGridView

Окремо розглянемо роботу форми створення нового завдання. При цій операції враховується безліч факторів а саме: тип робіт, що будуть виконуватися, доступність відповідних працівників, необхідної техніки та модулів для виконання цих робіт у вказані терміни. Після того як всі поля будуть заповнені можна буде створити нове завдання. Принцип роботи детальніше представлений на рисунку 3.18.

Після того як користувач успішно заповнить всі випадаючі списки, з’явиться можливість сформувати нове завдання, що буде включати ідентифікаційні значення всіх попередніх елементів. Таким чином пізніше ми зможемо переглядати усі завдання та розуміти їх деталі.

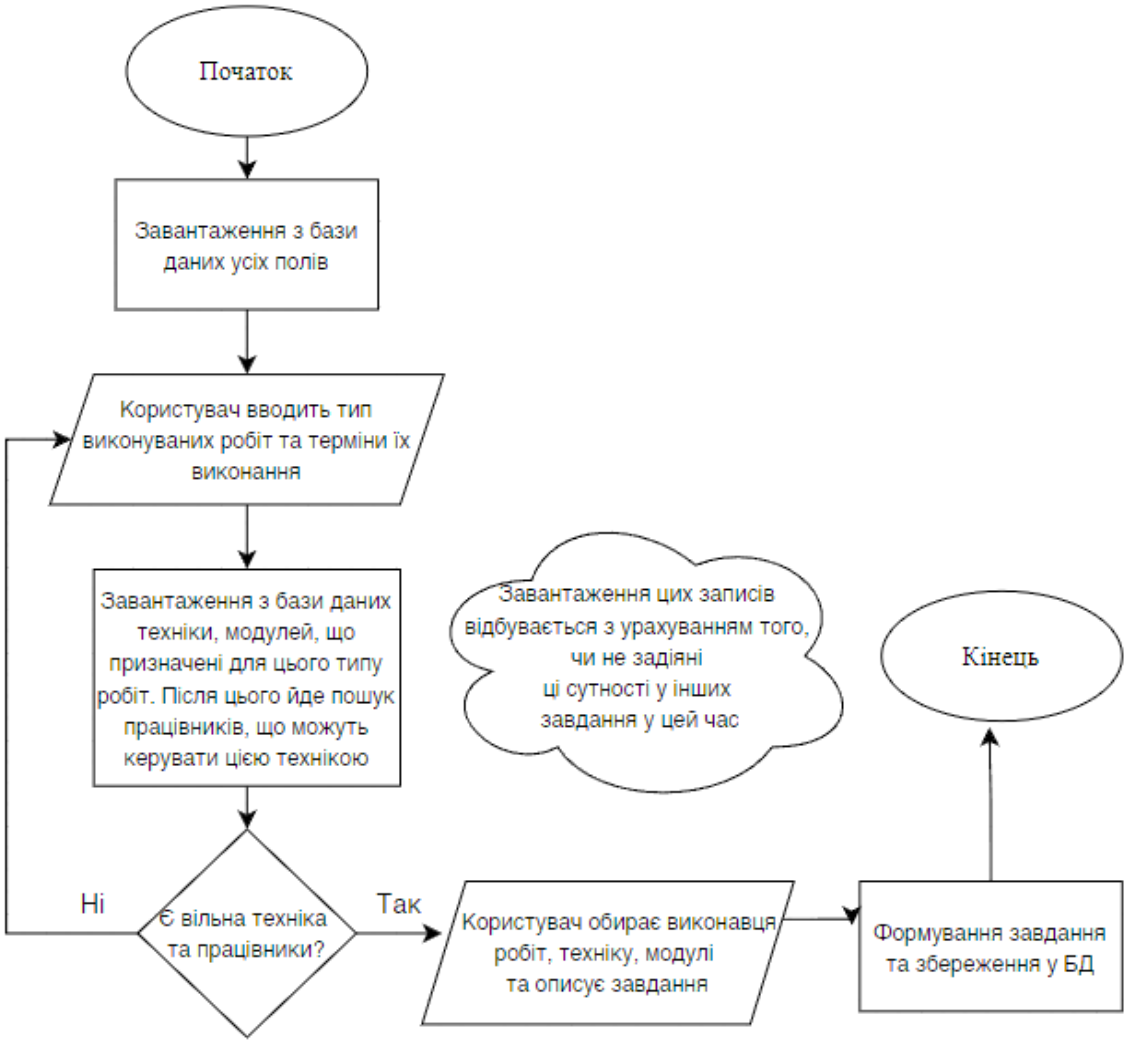

Рисунок 3.18 – Блок-схема форми створення нових завдань

Загалом, наш додаток вийшов вдалим. Ми створили добре спроектовану та оптимізовану базу даних, забезпечили приємний і зрозумілий інтерфейс користувача, і, найголовніше, реалізували весь запланований функціонал для планування аграрних робіт.

РОЗДІЛ 4. ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ ДОДАТКУ

4.1. Головні атрибути оцінювання якості та методологія тестування програмного забезпечення

Тестування — це процес оцінювання системи або її компонентів для перевірки відповідності вимогам чи виявлення дефектів [19]. Тестування відіграє ключову роль у забезпеченні якості програмного забезпечення, допомагаючи виявити та усунути помилки до випуску продукту. Це дозволяє зменшити ризики, підвищити задоволеність користувачів і забезпечити відповідність бізнес-вимогам.

До головних атрибутів оцінювання якості програмного забезпечення можна віднести наступні елементи:

- Функціональність – відповідність системи функціональним вимогам.
- Надійність – стійкість до помилок та стабільність роботи.
- Продуктивність – ефективність використання ресурсів системи.
- Зручність використання – легкість у взаємодії з інтерфейсом.
- Супроводжуваність – простота внесення змін та розширення функціоналу.
- Портативність – здатність працювати на різних платформах чи пристроях.

Методологія тестування – це система підходів, принципів та методів, які використовуються для перевірки програмного забезпечення з метою оцінки його якості. Розглянемо основні методології:

Ручне тестування – це процес перевірки програмного забезпечення, який виконується вручну без використання спеціальних інструментів автоматизації. Тестувальник самостійно виконує тестові сценарії, перевіряє функціональність, зручність використання та коректність роботи системи. Основною перевагою

ручного тестування є можливість оцінити поведінку програми з точки зору кінцевого користувача, тоді як недоліком є більший обсяг часу, необхідний для виконання тестів.

Автоматизоване тестування – це метод перевірки програмного забезпечення, який передбачає використання спеціальних інструментів і скриптів для виконання тестових сценаріїв. Воно дозволяє швидко та ефективно перевіряти великі обсяги даних, повторювати тести та проводити їх на різних конфігураціях. Автоматизація особливо корисна для регресійного тестування та перевірки складних систем, проте потребує значних початкових зусиль для розробки тестових сценаріїв і налаштування середовища.

4.2. Тестування розробленого десктопного додатку

Для нашого навчального проекту ми провели тестування методом "чорного ящика". Цей підхід фокусується на перевірці роботи додатку з точки зору користувача, без заглиблення в деталі реалізації коду. Тестування відбувається шляхом подачі вхідних даних і порівняння результатів з очікуваними. Цей метод ідеально підходить для навчальних проектів, оскільки дозволяє перевірити основний функціонал додатка, забезпечуючи достатню якість без складних процесів налаштування.

Для цього тесту ми перевіряли основний функціонал програми. Для тесту ми обрали вкладку "Працівники", яку ми вже попередньо розглядали

В першу чергу ми спробували посортувати наших працівників за певними критеріями, змінювали їх дані. Загалом все працює коректно, жодних помилок не виникає, інформація зберігається вірно.

Наступним кроком ми створили нового працівника з даними як на рисунку 4.1.

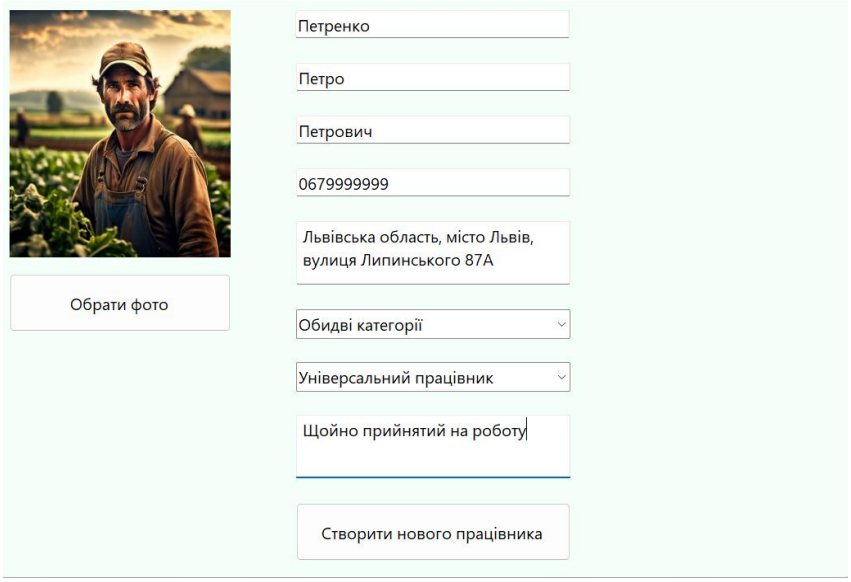

Рисунок 4.1 – Форма для запису даних нового працівника

У випадку, якщо одне з полів не заповнено або формат номеру телефону був неправильний то програма нас про це попереджає. Також якщо ми вказали, що працівник володіє категорією лише на трактор, то посаду комбайнера обрати неможливо. Після заповнення всіх даних новий запис був успішно збережений у БД.

Після цього ми створили кілька завдань для цього працівника. Для перевірки коректності цієї операції перевірили профіль на наявність новостворених записів.

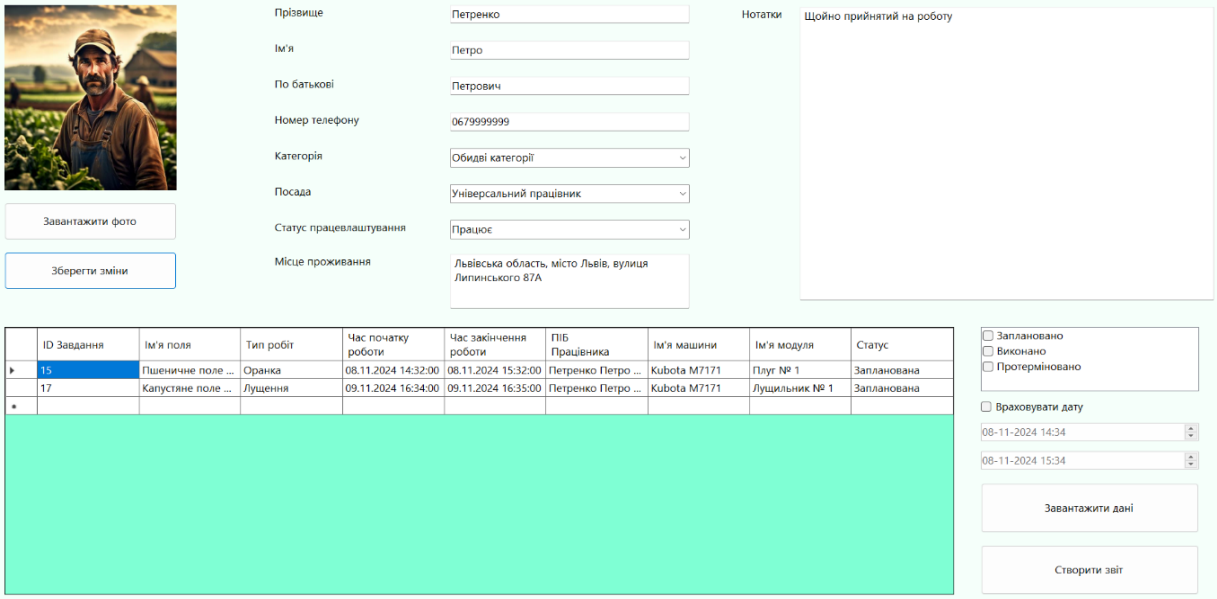

Рисунок 4.2 – Особистий кабінет і профіль працівника

Як бачимо на рисунку 4.2 уся інформація зберіглася коректно, нові завдання успішно завантажилися у профіль працівника. Також ми перевірили можливість змінити дані робітника та зберегти їх. Усе працює коректно. Створення PDF-звіту завдань теж працює належно. Є можливість відсортувати завдання за статусом виконання чи датою для більш зручного подання виконавцю.

Рисунок 4.3 – Перелік робіт для окремого працівника

Окремо перевірили роботу гістограми, що відображає всі роботи на конкретному полі за певний період. Для цього обрали одну земельну ділянку, вказали часові рамки та натиснули кнопку "Оновити дані".

Рисунок 4.4 – Гістограма виконання робіт на полі

Жодних помилок виявлено не було. Кожен тип робіт має унікальний колір, який коректно відображається, а часові рамки функціонують належним чином. Можна однозначно сказати, що це корисний додатковий інструмент для візуалізації даних і покращення їх розуміння.

Ми також перевірили всі форми додатка: CRUD-операції виконуються коректно, а фільтрація даних працює належним чином. Критичних помилок не виявлено. Інтерфейс виглядає приємно, меню інтуїтивно зрозуміле, і загальний досвід користування додатком залишає позитивне враження.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Структурно-функціональний аналіз процесу

Розробка та вживання ефективних заходів запобігання аварійним і травмонебезпечних ситуаціям можливі лише при завчасному виявлення тих небезпек, з яких починаються процеси їх формування. Оскільки небезпечні умови не завжди завчасно можна виявити, а для вивчення небезпечних дій іноді потрібно багато часу, щоб зібрати статичний матеріал, то і методи виявлення цих небезпек повинні бути відповідно диференційовані (табл. 5.1) [1].

Таблиця 5.1. Моделі формування та виникнення травмонебезпечних і аварійних ситуацій

Вид робіт, виробничий підрозділ, робоче місце, виробниче обладнання, склад агрегату	Виробнича небезпека			Можливі наслідки	Заходи запобігання небезпечним ситуаціям
	Небезпечна умова (НУ)	Небезпечна дія (НД)	Небезпечна ситуація (НС)		
Виконання робіт із електрообладнанням	Не вимкнено живлення. Відсутність заземлення.	Нехтування правилами ТБ	Ураження струмом	Травма (Т)	Проведення повторного інструктажу з ТБ. Розробка нових способів захисту. Встановлення заземлення.
НД НУ → НС → Т					

Відповідно до аналізу небезпечних умов, які існують у виробничому процесі виокремлено такі наступні за характером дії на працівника їх групи [1]:

– характеризують стан або рівень небезпеки обладнання, які використовуються.

– сприяють виникненню технологічних помилок обслуговуючого персоналу впродовж виробничого процесу;

– створювати умови та можливість проникнення працівника в небезпечну зону;

– приводять до виникнення небезпечних дій (внаслідок низького рівня професійної підготовки працівників та організації навчання з охорони праці).

Моделі формування та виникнення травмонебезпечних і аварійних ситуацій в комп'ютерному кабінеті представлено у вигляді моделі формування та виникнення травмонебезпечних і аварійних ситуацій – табл. 5.1.

5.2. Планування заходів із покращення умов праці

Освітленість виробничих приміщень може бути штучною і природною. Природне освітлення при правильному обладнанні найбільш сприятливе для людини. Основні вимоги для освітлення наступні:

– освітлення повинне бути достатнім для швидкого і легкого розпізнання об'єктів роботи;

– освітлення повинно бути рівномірне без різких тіней;

– джерело світла не повинно осліплювати працівника;

– рівень освітленості не повинен обмежуватись часом.

Природне освітлення забезпечується обладнанням вікон (бокове освітлення) фонарів і світильних покрить приміщень (верхнє освітлення). Природне освітлення нормується коефіцієнтом природної освітленості. Коефіцієнт природної освітленості – це процентне відношення фактичної освітленості $Fв$ в будь-якій точці приміщення до освітленості $Fн$ розсіяної світлом небозводу точки, яка лежить на відкритій місцевості. Розрахунок

природного освітлення через бокові вікна по нормам освітленості ведеться для самої дальньої від вікон точки, тобто знаходять мінімальне значення стик коефіцієнта природної освітленості [1]:

$$e_{\min} = \frac{Fb}{Fн} \cdot 100. \qquad (5.1)$$

Значення коефіцієнта природної освітленості визначається не менше чим в п'яти точках. Значення коефіцієнта природної освітленості для сільськогосподарських виробничих приміщень в даному випадку ремонтній майстерні, беремо $e_{min} = 5\%$.

Розрахунок природного освітлення зводиться до визначення площі світлових променів.

Сумарну площу світлових променів $\sum F_o \, (м^2)$ по коефіцієнту природної освітленості для бокових променів визначаємо по формулі:

$$\sum F_o = \frac{F_н \cdot e_{\min} \cdot r_o \cdot K}{100 \cdot \tau \cdot Г_1}, \qquad (5.2)$$

де $F_н$ – площа підлоги, м^2; e_{min} – величина мінімального коефіцієнта природного освітленості; τ – загальний коефіцієнт світловикористання віконного отвору із врахуванням його забруднення, τ = 0,25; r_o – світлова характеристика вікна, r_o – 9,5; $Г_1$ – коефіцієнт, який враховує підвищення освітленості за рахунок світла, яке відбивається від стін і стелі, $Г_1$ = 1,2; K – коефіцієнт, який враховує затінення вікон сусідніми приміщеннями і загорожею, K = 1.

$$\sum F_o = \frac{36 \cdot 0,5 \cdot 9,5 \cdot 1}{100 \cdot 0,25 \cdot 1,2} = 5,7 м^2.$$

Кількість світлових променів визначимо:

$$N = \frac{\sum F_o}{F_o}, \qquad (5.3)$$

де F_o – площа вікна згідно стандарту, м^2.

$$N = \frac{5,7}{6} = 0,95.$$

Приймаємо кількість вікон – одне вікно.

При розрахунку природного освітлення найбільш поширеним і простим є метод світлового потоку. При цьому методі розраховуємо світловий потік $F_л$ (Лк), який повинна випромінювати кожна лампа (при заданій кількості ламп).

$$F_л = \frac{k \cdot S_п \cdot E}{n_л \cdot \eta \cdot r^2}, \qquad (5.4)$$

де k – коефіцієнт запасу, $k = 1{,}3$; $S_п$ – площа підлоги, м^2; $S_п = 36$ м^2. E – нормативна освітленість, Е = 300 Лк; $n_л$ – кількість встановлених ламп, $n_л$ =6 од; η – коефіцієнт використання світлового потоку, $\eta = 0{,}25$; r – коефіцієнт нерівномірності освітленості, $r = 0{,}545$.

Коефіцієнт запасу (K) враховує можливість забруднення світильників пилом, що залежить від характеру виробництва.

Розрахунок штучного освітлення починаємо із визначення висоти розташування світильника і їх кількості. Висоту $h_н$ (м) розташування світильників над робочим місцем знаходимо за формулою:

$$h_н = H - (h_1 + h_2), \qquad (5.5)$$

де H – висота приміщення, м; h_1 – віддаль від підлоги до освітлювальної поверхні, м; h_2 – віддаль від стелі до світильника, м.

$$h_н = 4{,}5 - (2{,}2 + 1{,}5) = 0{,}8 \text{ м}.$$

При симетричному розміщенні світильників по вершинах квадратів їх кількість визначається за формулою:

$$n_c = \frac{S_п}{l^2}, \qquad (5.6)$$

де l – віддаль між світильниками, м.

Підставивши значення отримаємо:

$$n_c = \frac{36}{9} = 4 \text{ од}.$$

Тоді світловий потік буде становити:

$$F_л = \frac{1,3 \cdot 36 \cdot 300}{4 \cdot 0,25 \cdot 0,545} = 2576,2 \text{ Лк}.$$

При світловому потоці 2576,2 Лк для заданої лампи вибираємо тип і потужність.

Вибираємо тип лампи – люмінесцентну, потужністю 40Вт.

5.3. Безпека в надзвичайних ситуаціях

Забезпечення захисту населення і території у разі загрози і виникнення надзвичайних ситуацій є одним з най важливіших завдань держави.

Захист населення є системою загально державних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами влад, органами управління з питань надзвичайних ситуацій та цивільного захисту населення, підпорядкованими їм системами, та підприємств, що забезпечують виконання організаційних, інженерно – технічних, санітарно – гігієнічних, проти епідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєво важливих інтересів громадян, держави, суспільства поділяють на зовнішні та внутрішні, виконують під час надзвичайних ситуацій техногенного та природного характеру та воєнних конфліктах.

Принципи захисту випливають з основних положень Женевської конвенції щодо захисту жертв війни та додаткових протоколів до неї, можливого характеру воєнних дій, реальних можливостей держави щодо створення матеріальної бази захисту. З метою захисту населення, зменшення втрат та шкоди економіці в разі виникнення надзвичайних ситуацій має право проводитись спеціальний комплекс заходів.

Оповіщення та інформування, яке досягається завчасним створенням і підтримкою в постійній готовності загально державної, територіальних та об’єктивних систем оповіщення населення.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У процесі роботи ми проаналізували сучасні тенденції в технологіях управління земельними ресурсами. Зокрема, вивчили концепцію "Точного землеробства" та методи, що сприяють реалізації цієї технології в аграрному менеджменті. Ми розглянули застосування GPS, види географічних інформаційних систем, а також використання IoT і штучного інтелекту в агробізнесі.

Ми також дослідили популярне програмне забезпечення для агробізнесу. Виявилося, що більшість таких рішень багатофункціональні, але мають свої недоліки. Чимало з них включають опції, які можуть бути недоступні для певних фермерів, як-от сумісність лише з технікою конкретних марок. Більшість таких додатків існують лише в мобільних або веб-версіях, що залишає нішу для десктопних програм. Тому ми вирішили створити настільний додаток для зручного та інтуїтивного агропланування.

У наступному розділі ми дослідили концепцію настільного додатка, його основні типи та порівняли з веб-додатками для глибшого розуміння переваг і недоліків класичних додатків. Розглянули популярні операційні системи, проаналізували їхні особливості та можливості розробки нативних і кросплатформних додатків. Обрали Windows, враховуючи значну аудиторію, що використовує її для бізнесу. Серед фреймворків зупинилися на Windows Forms через простоту розробки й низький поріг входження, що дозволило зосередитися на бізнес-логіці програми.

У розділі розробки ми створили концептуальну модель нашого додатку, базуючись на власних спостереженнях і досвіді з аналогічними програмами, що допомогло сформувати унікальне бачення і підхід до реалізації. Також було розроблено попередні макети вікон користувача для кращого розуміння їхньої майбутньої взаємодії з інтерфейсом. Після завершення етапу планування ми розпочали практичну реалізацію.

Спершу ми розробили базу даних. Для кожної сутності була створена таблиця з відповідними полями. Остання таблиця призначена для зберігання завдань для працівників і містить інформацію про поле, де проводяться роботи, тип робіт, терміни, необхідну техніку та модулі до неї, а також виконавця завдання.

Наступним кроком було створення форм для програми за допомогою візуального редактора та їх програмування. Основний функціонал розробляли у функціональному стилі, що дозволило швидко реалізувати всі заплановані можливості без глибокого використання інших програмних патернів.

Завершальним етапом стало тестування методом "чорного ящика". Перевірка показала, що всі модулі працюють стабільно без критичних помилок, інтерфейс зручний та інтуїтивний, а користування програмою залишає позитивне враження.

Загалом, нам вдалося успішно реалізувати всі цілі, поставлені в кваліфікаційній роботі. Розроблений додаток працює стабільно в ОС Windows, має приємний та інтуїтивно зрозумілий інтерфейс і, що найважливіше, виконує всі заплановані функції. Це вузькоспеціалізований інструмент для планування аграрних робіт, що ефективно відповідає вимогам та допомагає користувачам організувати свої задачі у відповідному контексті.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Запорожець О. В. Протоєрейський О. П., Франчук Г. І., Боровик І. В. Основи охорони праці: 2019 р. 264 с.

2. Клімат України [За редакцією В.М. Ліпінського, В.А. Дячука, В.М. Бабіченко]. Київ: видавництво Раєвського, 2003. 344 с.

3. Крокфорд. Д. Як влаштований JavaScript: Навчальний пос. / Д. Крокфорд, К .: Міннесота, 2019. 304 с.

4. Прусов В.А., Дорошенко А.Ю. Моделювання природних і техногенних процесів в атмосфері. – К.: Наукова думка, 2006. – 542 с.

5. Переваги та недоліки використання хмарних технологій підприємствами України. URL: http://www.bsfa.edu.ua/files/konf2013/62.pdf (дата звернення: 20.04.2025).

6. Стеценко І.В. Моделювання систем: навч. посіб. [Електронний ресурс, текст] / І.В. Стеценко ; М-во освіти і науки України, Черкас. держ. технол. ун-т. Черкаси : ЧДТУ, 2020. 399 с.

7. Тимченко А.А. Основи системного проектування та системного аналізу складних об’єктів: Підручник для студентів вищих закладів освіи / За ред. В.І. Бикова. К.: Либідь, 2010. 270 с.

8. Томашевський В. М. Моделювання систем. К.: Видавнича група BHV, 2015, 352 с.

9. ТОП сервісів спостереження за погодними умовами. URL: https://kr-labs.com.ua/blog/top-weather-forecast-services (дата звернення 04.04.2025).

10. Agrivi. URL: https://www.agrivi.com/products/360-farm-enterprise/ (дата звернення 04.04.2025).

11. Folio3 AgTech. URL: https://agtech.folio3.com/blogs/best-crop-management-software-solutions/ (дата звернення 28.05.2025).

12. Fortune business insights. URL: https://www.fortunebusinessinsights.com/farm-management-software-market-110388 (дата звернення 25.05.2025).

13. GlobalX. URL: https://globalx-ua.com/internet-veschey-v-selskom-hozyaystve (дата звернення 29.05.2025).

14. Granneman, Scott. Linux Phrasebook (Developer's Library). 2nd Edition. 2019. 464 p.

15. GTK. URL: https://www.gtk.org/ (дата звернення 27.05.2025).

16. Infopulse. URL: https://www.infopulse.com/blog/software-solutions-agriculture (дата звернення 27.05.2025).

17. Learn Microsoft .NET MAUI. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-9.0 (дата звернення 01.05.2025).

18. Learn Microsoft Windows Forms. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-9.0 (дата звернення 02.05.2025).

19. Microsoft Visual Studio. URL: https://visualstudio.microsoft.com/ (дата звернення 05.10.2024).

20. Petzold, Charles. Programming Windows. Fifth Edition (Microsoft Programming Series). 2002. 1096 p.

21. Richter, Jeffrey. CLR via C#. 4th Edition. 2012. 896 p.

ДОДАТКИ

Додаток А
Фрагменти коду десктопного додатку

```
DataBase db = new DataBase();
5 references
public Employees_Form()
...
1 reference
private void CreateItemsForCheckedListBox() //Додає айтеми сортування до checkedListBox //
...
1 reference
private void checkedListBox1_ItemCheck(object sender, ItemCheckEventArgs e) // Не дозволяє вибирати більше одного
                                                                             // айтема checkedListBox з перших трьох але дозволяє обирати четвертий
...
1 reference
private void CreateColumnsForDataGridView() // Створює колонки dataGridView (1 раз) та розширює всі колонки заповнюючи весь доступний простір
...
1 reference
private static string CategoryCheck(int value) // Перетворює числове значення з БД в строкове значення з назвою Категорії
...
1 reference
private static string JobTitleCheck(int value) // Перетворює числове значення з БД в строкове значення з назвою Посади
...
1 reference
public int UnemploymentItemCheck() // Перевіряє чи обрано айтем звільнений та повертає int значення для запиту в БД
...
1 reference
private void ReadSingleRow(DataGridView dgv, IDataRecord record)
...
1 reference
private string GetQuery()
...
3 references
public void RefreshDataGridView(DataGridView dgv) // Оновлює DataGridView
...
1 reference
private void button1_Click(object sender, EventArgs e) //Кнопка оновити дані
```

Рисунок А.1 - Методи форми "Працівники"

```
private void button_Add_new_Employee_Click(object sender, EventArgs e) //Відкриває вікно редагування
...

/////////////// Редагування даних ///////////////

public int? selectedEmployeeID; // Зберігає значення поточно вибраного id
1 reference
private void dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
...
1 reference
private bool CheckFields() // Перевіряє чи всі поля для редагування працівників заповнені
...
1 reference
private void MakeTextBox() // Плейсхолдери для textboxів, назви для comboxів та айтеми для comboBox_license_category
...
1 reference
private bool GetCategoryCompatibility()
...
0 references
private void comboBox_license_category_SelectedIndexChanged(object sender, EventArgs e)
...
1 reference
private void button_Save_Changes_Click(object sender, EventArgs e)
...
1 reference
private void ClearUpdateFileds()
...
1 reference
private void dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e) // При подвійному кліку по рядку відкриває вікно працівника
...
1 reference
... // Управління кнопками
```

Рисунок А.2 - Методи форми "Працівники"