МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **“ Розробка десктопного додатку на базі API Windows Forms для координації польових робіт у сільськогосподарському підприємстві ”**

Виконав: __ст. гр. Іт-61_____

Спеціальності 126 – «Інформаційні системи та технології»_____
(шифр і назва)

_____Леськів Назарій Романович_____
(Прізвище та ініціали)

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

Рецензент: _____
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

другий (магістерський) рівень вищої освіти
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Леськіву Назарію Романовичу

1. Тема роботи: «Розробка десктопного додатку на базі API Windows Forms для координації польових робіт у сільськогосподарському підприємстві»

Керівник роботи Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету від 12.09.2024 № 616/к-с.

2. Строк подання студентом роботи - 06.12.2024.

3. Початкові дані до роботи: 1. Технічна і довідкова література. 2. Інформаційні ресурси розробників мов програмування. 3. Сегментація ринку — аграрний сектор. 3. Особливості синтаксису C#. 4. Стандарти роботи з Windows Forms. 5. Інтеграція баз даних у десктопних застосунках.
4. Зміст розрахунково-пояснювальної записки:

1. *Аналіз сучасного стану використання IT-сервісів в агровиробництві*
2. *Інструменти для створення десктопного додатку в системі Windows*
3. *Розробка десктопного додатку на базі 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 – Актуальні технології в агробізнесі; 4 – Аналіз застосунків для агропланування; 5 – Поняття десктопного застосунка; 6 – Огляд платформ; 7 - Огляд технологій Windows Forms, WPF, .NET MAUI; 8 - Створення концепції; 9 - Розробка десктопного додатку; 10 - Тестування; 11 - Демонстрація роботи; 10 - Головні висновки.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4, 6	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання – 12 вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк вик. роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	12.09 - 01.10.24	
2.	*Виконання другого розділу та опис інформаційних технологій для виконання завдань роботи*	12.09 - 01.10.24	
3.	*Виконання третього розділу, методика вирішення завдань та елементи наукових досліджень*	01.10 - 01.11.24	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	01.10 - 01.11.24	
5.	*Головні результати отримані в роботі, оцінення розроблених пропозицій*	01.11 - 01.12.24	
6.	*Завершення оформлення розрахунково-пояснювальної записки та презентаційних матеріалів*	01.11 - 01.12.24	
7.	*Завершення роботи в цілому*	01-10.12.24	

Магістрант _____ Леськів Н.Р.
(підпис)

Керівник роботи_____ Луб П.М.
(підпис)

УДК: 004.414.23:631.3.023.6

Кваліфікаційна робота: 72 с. текст. част., 33 рис., 2 табл., 12 слайдів, 28 джерел.

Розробка десктопного додатку на базі API Windows Forms для координації польових робіт у сільськогосподарському підприємстві. Леськів Н.Р. Кафедра ІТ. – Дубляни, Львівський НУП, 2024.

Проаналізовано сучасний стан використання інформаційних технологій в аграрному секторі, зокрема тенденції у сфері управління земельними ресурсами. Розглянуто програмні рішення для агропланування, а також сформульовано задачу створення настільного додатка для планування агротехнічних робіт.

Досліджено концепцію десктопних застосунків та їхні різновиди. Проведено огляд популярних операційних систем і фреймворків для розробки настільних застосунків.

Розглянуто інструментарій технології Window Forms. Розроблено концептуальну модель програми для оптимізації агропланування. Розроблено десктопний застосунок враховуючи всі проаналізовані фактори.

Проаналізовано ключові критерії оцінки якості програмного забезпечення та методології його тестування. У межах роботи проведено тестування розробленого настільного додатка, що підтвердило його відповідність поставленим вимогам.

Означено вимоги з охорони праці та безпеки в надзвичайних ситуаціях.

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

ЗМІСТ

Зміст

ВСТУП

Сучасну агроіндустрію неможливо уявити без спеціалізованого програмного забезпечення. Спеціалізовані додатки допомагають автоматизувати процеси, аналізувати різноманітні дані, організовувати працю робітників, керувати технікою та виконувати багато інших завдань. Завдяки найсучаснішій техніці та спеціальному ПЗ вдалося організувати всі процеси найвдалішим чином, що дозволяє виконувати велику кількість робіт при незначних ресурсах.

Кожен фермер може знайти для себе вдале рішення, що дозволить ефективно проаналізувати та спланувати роботи від підготовки ґрунту до збирання врожаю. Використання таких інструментів підвищує продуктивність, знижує витрати і сприяє більш точному управлінню всіма аспектами сільськогосподарських операцій, забезпечуючи стабільний розвиток господарства.

Головним завданням розроблюваного нами додатку буде зручне та зрозуміле планування агро-робіт для невеликих підприємств. Оскільки більшість агро-додатків є багатофункціональними, це часто призводить до складності їх використання і вимагає значних зусиль для навчання. Наш додаток, навпаки, орієнтований на конкретні задачі і простоту використання, що дозволяє фермерам-початківцям та власникам невеликої кількості полів швидко адаптуватися та ефективно управляти своїми господарствами.

Завдяки інтуїтивно зрозумілому інтерфейсу та зосередженості на основних функціях, таких як планування польових робіт та відслідковування статусів їх виконання наш додаток допоможе фермерам максимально ефективно організувати свою роботу. Це не лише спростить процес управління господарством, але й дозволить зосередитися на важливих аспектах агробізнесу, таких як підвищення врожайності та оптимізація ресурсів, забезпечуючи стабільний розвиток та успіх у довгостроковій перспективі.

Об’єктом роботи є процес розробки десктопних додатків.

Предметом роботи є створення десктопного додатку для координації польових робіт у сільськогосподарському підприємстві на базі API Windows Forms.

Актуальність даного проекту полягає у забезпеченні власників невеликих агропідприємств спеціальним програмним забезпеченням, що дозволить оптимізувати процес планування координації польових робіт.

Метою роботи є створення десктопного додатку для координації польових робіт у сільськогосподарському підприємстві на базі API Windows Forms.

Завдання роботи – створити десктопний додаток для координації польових робіт у сільськогосподарському підприємстві шляхом вирішення наступних завдань:

- проаналізувати ринок актуальних агро-планувальників;
- розкрити особливості, вибрати інструментальні засоби та технологію для розробки десктопного додатку щодо координації польових робіт у сільськогосподарському підприємстві;
- описати концепцію реалізації десктопного додатку на базі API Windows Forms;
- розробити додаток для координації польових робіт у сільськогосподарському підприємстві.

Практична цінність роботи полягає в наступних аспектах:

- Освоєння технологій: Розробка десктопного додатку на базі API Windows Forms дозволяє студенту поглибити свої знання та навички у галузі програмування та розробки програмного забезпечення для сільськогосподарських підприємств.
- Розширення технічних навичок: Робота з Windows Forms надає можливість вивчити і використовувати різноманітні функції та інструменти, такі як створення інтерфейсу користувача, управління базами даних, обробка подій та інші компоненти, що розширює багаж технічних навичок студента.
- Проектування та управління процесом розробки: Розробка десктопного додатку вимагає створення концепції, дизайну інтерфейсу,

розробки функціоналу та тестування. Це дозволяє студенту отримати досвід у проектуванні та управлінні процесом розробки програмного забезпечення.

- Творчість та самовираження: Розробка власного десктопного додатку дає можливість втілити свої ідеї, творчість та власний стиль у програмі. Це дозволяє розкрити свій потенціал як розробника і виразити себе через створений додаток.
- Практичний досвід та портфоліо: Результатом дипломної роботи є готовий десктопний додаток для координації польових робіт у сільськогосподарському підприємстві, який можна включити до свого портфоліо. Це дає змогу демонструвати свої навички та досягнення потенційним роботодавцям у галузі розробки програмного забезпечення.
- Задоволення користувачів: Кінцевим результатом розробки додатку є задоволення користувачів, які зможуть ефективно організовувати та керувати польовими роботами. Практична цінність полягає у тому, що ми маємо можливість створити корисний продукт, який може значно полегшити управління сільськогосподарськими операціями.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ВИКОРИСТАННЯ ІТ-СЕРВІСІВ В АГРОВИРОБНИЦТВІ

1.1 Тенденції інформаційних технологій із управління земельними ресурсами

Кожна галузь зазнала позитивного впливу ІТ технологій, і сільське господарство не є винятком. Ферми та сільськогосподарські об'єкти прагнуть використовують передові методи землеробства, відомі також як точне землеробство. Цінність, яку точне землеробство приносить в управлінні полями та врожаєм, величезна. Підвищення прибутку, ефективність, безпека та простота - це нова норма, яка, безсумнівно, рухатиме галузь вперед у найближчому майбутньому.

Точне землеробство - комплексна високотехнологічна система сільськогосподарського менеджменту, що включає в себе технології глобального позиціонування (GPS), географічні інформаційні системи (ГІС), інтернет речей (IoT), технології оцінки врожайності та змінного нормування на основі даних цих технологій [6]. Метою такого підходу звичайно є підвищення якості та кількості сільськогосподарської продукції при одночасній оптимізації людської праці для забезпечення найкращих можливих результатів.

Впровадження технології GPS здійснило революцію в сільському господарстві, уможлививши точне позиціонування та навігаційні системи для сільськогосподарської техніки. Цей прорив дав фермерам можливість ефективніше виконувати завдання на полях, як-от посадка насіння та точніше розпилення добрив. Завдяки GPS, техніка може автоматично дотримуватися заданих маршрутів з високою точністю, що зменшує перекриття та пропуски, знижуючи витрати на паливо, добрива та інші ресурси. Крім того, системи GPS сприяють точному моніторингу та картуванню полів, допомагаючи виявляти проблемні зони та планувати оптимальне використання ресурсів. Це також дає

можливість збирати та аналізувати дані про врожайність з різних ділянок, що є важливою частиною прийняття рішень для підвищення ефективності виробництва.

Іншою важливою тенденцією є використання безпілотних літальних апаратів та супутникових знімків для моніторингу стану полів. Сільськогосподарські дрони зробили значний крок вперед за останні роки. По-перше, безпілотні технології стали набагато доступнішими, практичнішими та ефективнішими. Так, фермери тепер можуть використовувати сільськогосподарські дрони для огляду місцевості з висоти пташиного польоту та швидкого отримання візуальної інформації про конкретну ділянку. Отримавши високоточні знімки полів у реальному часі, з'являється можливість виявляти проблемні зони, наприклад, ділянки з недостатнім поливом або ураженням шкідниками. Модернізовані сільські господарства значною мірою покладаються на великі та точні набори даних, щоб розробити найкращий курс дій. Супутникові знімки, в свою чергу, забезпечують ширшу картину і можуть використовуватися для довгострокового моніторингу [12].

Рисунок 1.1 - Агродрон у роботі

Також не можна не згадати про вклад у точне господарство географічних інформаційних системи. ГІС - це інструмент, який дозволяє користувачам

створювати багатошарові інтерактивні карти, які можна використовувати для візуалізації складних даних і просторового аналізу [5]. Польова техніка на землі, дрони і супутники про які ми розповідали вище збирають сільськогосподарські дані, а згодом ця інформація може бути використана для широкого спектру цілей. В результаті ви отримуєте карту, яка показує місцезнаходження та рельєф ваших ділянок.

Слід зазначити, що ГІС у сільському господарстві може відрізнятися залежно від їх призначення. Деякі інструменти показують типи сільськогосподарських культур, стан посівів, врожайність. Інші візуалізують швидкість руху вітру, рівень вологості ґрунту, його температуру та pH. Використовуючи ці дані, фермери можуть визначати оптимальний час для посіву, збору врожаю та отримувати багато іншої корисної інформації.

Технологія інтернет речей принесла значні зміни в аграрне виробництво та залишається ключовою у цій сфері. Це рішення дозволяє підключати різного роду датчики та пристрої до єдиної системи, що дозволяє збирати і аналізувати різноманітні дані. Одним з основних застосувань IoT в агробізнесі є контроль за умовами росту та розвитку рослин. За допомогою сенсорів, встановлених у ґрунті, можна отримати дані про рівень вологості, температуру та поживні речовини. Це дозволяє аграріям забезпечувати оптимальні умови для росту рослин та своєчасно виявляти й вирішувати проблеми зі здоров'ям рослин.

Говорячи про технологічні тренди в програмному забезпеченні для аграріїв, тут відбувається слідування сучасним тенденціям, а саме використання: веб-сервісів, мобільних додатків, хмарних технологій для доступу до даних з будь-якого пристрою.

Веб-сервіси поступово витісняють традиційну розробку десктопних додатків завдяки своїй зручності та ефективності. Вони доступні з будь-якого пристрою з підключенням до інтернету, що забезпечує гнучкість та мобільність для користувачів.

Мобільні додатки пропонують значні переваги для сільського господарства, особливо працівників, які працюють безпосередньо в полі. За допомогою

смартфонів агрономи можуть в реальному часі перевіряти стан полів, отримувати оновлення про погодні умови, вводити дані про польові роботи та приймати оперативні рішення [24].

Хмарні технології відіграють ключову роль у сучасному агробізнесі, забезпечуючи централізоване зберігання та обробку даних, доступ до яких можливий з будь-якого пристрою, підключеного до Інтернету. Вони дозволяють зберігати великі обсяги інформації про стан полів, погодні умови, врожайність та інші критичні показники. Інтеграція хмарних сервісів з мобільними та веб-додатками забезпечує безперервний доступ до даних у реальному часі, що значно полегшує обмін інформацією між різними користувачами та їх пристроями [25].

Аналізуючи ринок програмного забезпечення для агробізнесу, можна відслідкувати одну важливу рису: створення комплексних рішень, що покривають всі потреби бізнесу [8]. Таке програмне забезпечення включає логістичні функції, планування робіт, аналіз стану врожаю, можливості прогнозування, фінансові розрахунки та звітність, управління персоналом і технікою. Водночас, вузькоспеціалізовані додатки також здобувають популярність. Як правило вони мають успіх через недосконаліть окремих модулів повністю готових рішень. Ці додатки фокусуються на конкретних задачах та потребах фермерів, надаючи більш точні та ефективні інструменти для вирішення специфічних проблем.

Одним з різновидів вузькоспеціалізованого аграрного забезпечення є агропланувальники, що є критично важливим для оптимізації використання ресурсів, своєчасного виконання сільськогосподарських завдань та підвищення загальної продуктивності. У цьому контексті виникає необхідність у розробці спеціалізованого агропланувальника, який буде не лише інтегрувати сучасні технології, але й бути зручним у використанні для аграріїв.

1.2 Аналіз застосунків для агропланування

Як було зазначено вище, всі агропланувальники зазвичай є складовими

комплексного аграрного ПЗ. Шукаючи в інтернеті рішення для агропланування, майже неможливо знайти додаток, який зосереджений виключно на цій функції. Тому серед наявних аналогів ми обрали ті, що користуються попитом серед фермерів, базуючись на статистиці завантажень у магазинах мобільних додатків та рейтингах на популярних спеціалізованих сайтах для фермерів [7].

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 [14].

Доступ до функцій платформи є безкоштовним, і навіть не потрібно бути власником техніки даної марки. Це робить John Deere Operations Center доступним і корисним для фермерів, які використовують техніку інших марок хоча варто враховувати, що функціонал при цьому може бути неповним.

Варто відмітити інтеграцію з багатьма іншими системами, що дозволяє швидко та зручно переносити дані.

Операційний центр складається з чотирьох основних функцій:

- Налаштування ферми. Можливість призначати членів команди, визначати поля та техніку для кращої структури бізнес процесів.
- Планування роботи. Автоматизація та керування роботою в будь-який час і в будь-якому місці. Інструменти для планування допомагають оптимізувати розклад робіт, забезпечуючи своєчасне виконання всіх агротехнічних заходів.
- Моніторинг польових робіт. Можливість віддалено оцінювати роботу обладнання та продуктивність в режимі реального часу для швидкого реагування на будь-які зміни.
- Аналіз даних. Аналітичні інструменти дозволяють детально аналізувати дані, отримані з полів, та приймати обґрунтовані рішення щодо управління господарством.

Враховуючи той факт, що у цій програмі безліч модулів і функцій ми зосередимо увагу на агропланування.

Планувальник в Operation Center дозволяє оперативно створювати плани робіт, що забезпечує гнучкість та зручність для фермерів. Плани можна створювати завчасно або в будь-який час за допомогою веб-версії чи мобільного додатку.

Існує можливість сортувати роботи за різними критеріями, такими як обробка ґрунту, посів, внесення добрив тощо. В залежності від вибраного типу робіт необхідно вибрати певну техніку, модулі для неї та інше спорядження.

Інструмент також має додаткові опції для деяких видів робіт. Для прикладу можна планувати посадку культур, враховуючи попередні посіви. Якщо на певних полях у попередньому році була посаджена кукурудза, система може рекомендувати посадку іншої культури у поточному році, враховуючи принципи сівозміни. Це допомагає покращити якість ґрунту та підвищити врожайність на окремих ділянках.

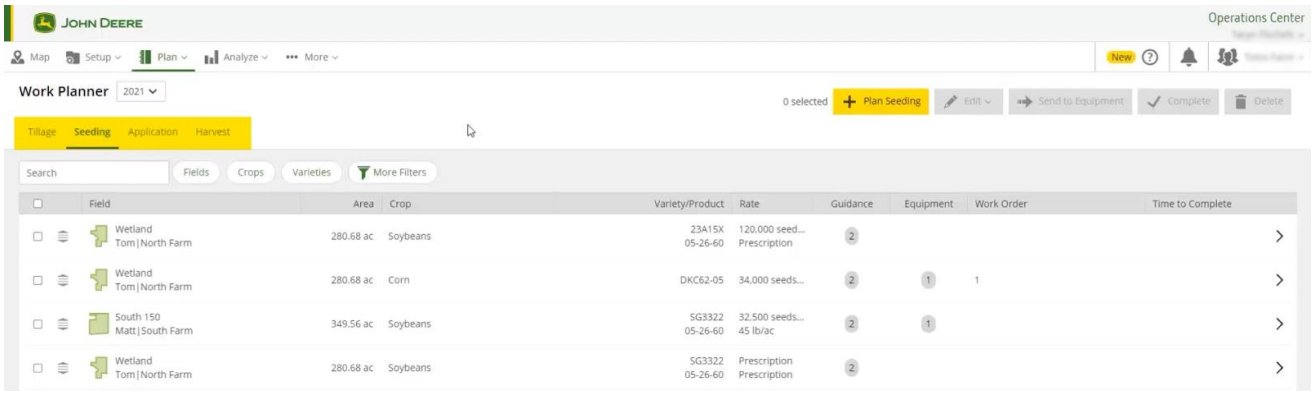

Рисунок 1.2 - Меню завдань платформи John Deere

Якщо ми вирішили посадити сою, можна обрати та встановити норми для посіву та внесення добрив, спеціально для цієї культури. Планувальник дозволяє легко встановити одну цільову норму для всіх полів, а потім, за потреби, відрегулювати її для окремих ділянок.

Плани робіт та навігаційні маршрути, створені в Operation Center автоматично надсилаються на обладнання техніки через бездротове з'єднання

JD-Link. Це забезпечує миттєвий доступ до планів на дисплеї машини, коли та приступає до роботи.

До плану робіт можна додати додаткові відомості, такі як номери нарядів на виконання робіт та інструкції, що допомагає забезпечити точне виконання завдань відповідно до плану.

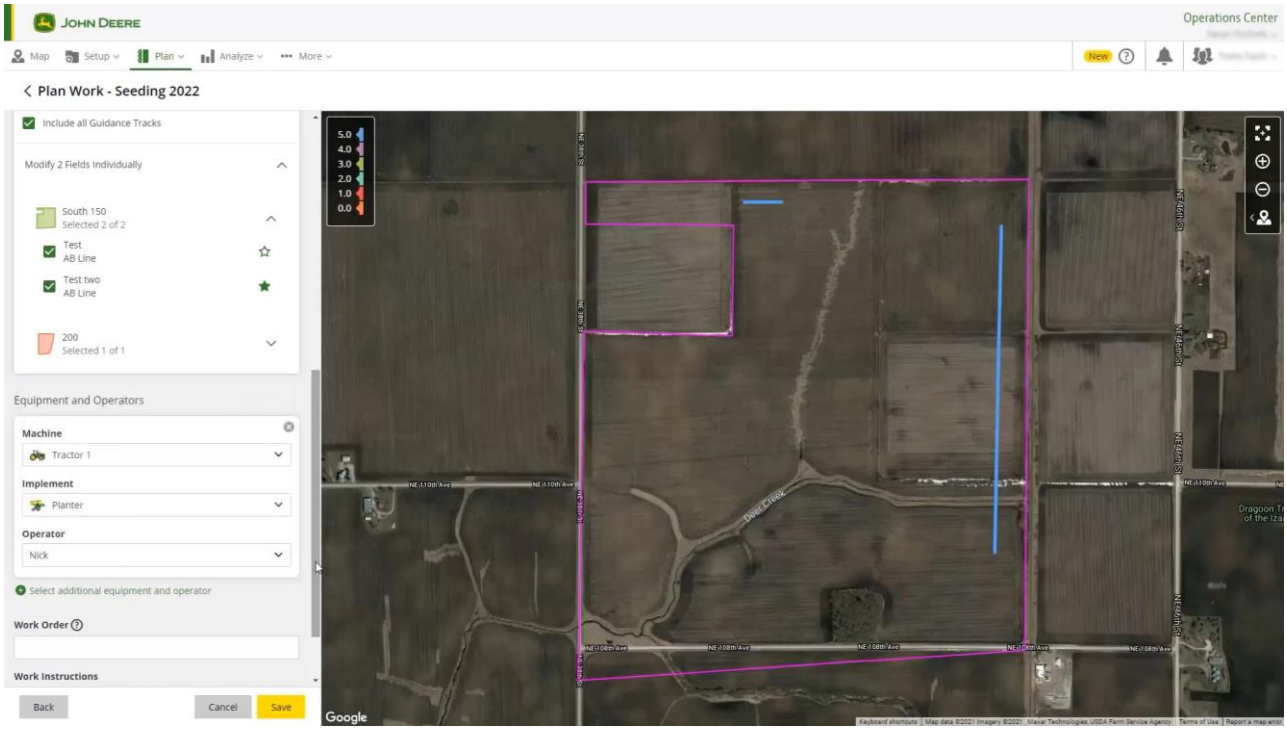

Рисунок 1.3 - Вікно планування задач

John Deere Operations Center - це інноваційне рішення, яке допомагає фермерам підвищити продуктивність, ефективність та прибутковість свого господарства. Незалежно від того, чи використовуєте ви техніку John Deere, ця платформа надасть вам необхідні інструменти для успішного управління сільськогосподарськими операціями.

Наступним на огляді буде Trimble Ag Software - це комплексне програмне забезпечення для управління сільськогосподарськими операціями, яке пропонує широкий спектр функцій для фермерів і агрономів. Платформа допомагає оптимізувати процеси управління полями, підвищувати врожайність та знижувати витрати [26].

Компанія Trimble спеціалізується на розробці різноманітних датчиків, систем автопілотів, спеціалізованих дисплеїв для техніки, а також програмного забезпечення для ефективного управління цими продуктами.

Основні функції Trimble Ag Software:

- Картографія та ГІС. Створення та управління картами полів з використанням геоінформаційних систем. Це включає моніторинг ґрунтів, аналіз врожайності та інші просторові дані.
- Планування польових робіт. Інструменти для планування посівів, обробки ґрунту, внесення добрив та інших агротехнічних заходів.
- Моніторинг техніки. Відстеження роботи сільськогосподарської техніки в реальному часі, включаючи збір даних про продуктивність і стан обладнання. Це допомагає вчасно виявляти та усувати проблеми.
- Аналіз та звітність. Інструменти для аналізу даних та генерування звітів про врожайність, витрати, ефективність робіт тощо.

Також варто відзначити чудову інтеграцію для обміну даними з іншим сільськогосподарським програмним забезпеченням, таким як Raven Slingshot, системи AGCO VarioDoc і AgCommand, John Deere Operations Center та багатьма іншими.

Програмне забезпечення дозволяє централізовано зберігати всі дані про сільськогосподарські поля, що спрощує управління інформацією та забезпечує доступ до актуальних даних у будь-який час. Завдяки інтуїтивно зрозумілому інтерфейсу, фермери можуть легко керувати своїм бізнесом, не відволікаючись на рутинні завдання.

Важливо розуміти, що весь функціонал ПЗ тісно пов’язаний з обладнанням компанії і без нього працювати програма не буде. Тому фермери повинні бути готовими до додаткових витрат.

Функція Робочі завдання призначена для оптимізації робочих процесів і являє собою набір вказівок щодо виконання польових робіт. Завдання створюються через Інтернет, а потім синхронізуються з фірмовими дисплеями компанії, щоб спростити їх віддалене налаштування. Це дає змогу працівникам

мати детальну інформацію про матеріали, поля, агрегати та операторів на дисплеї, ще до того, як вони сядуть у кабіну. Після того, як оператор завершить виконання завдання за допомогою дисплея, стан завдання змінюється, а дані робочого завдання бездротовим зв'язком передаються в програмне забезпечення Trimble Ag.

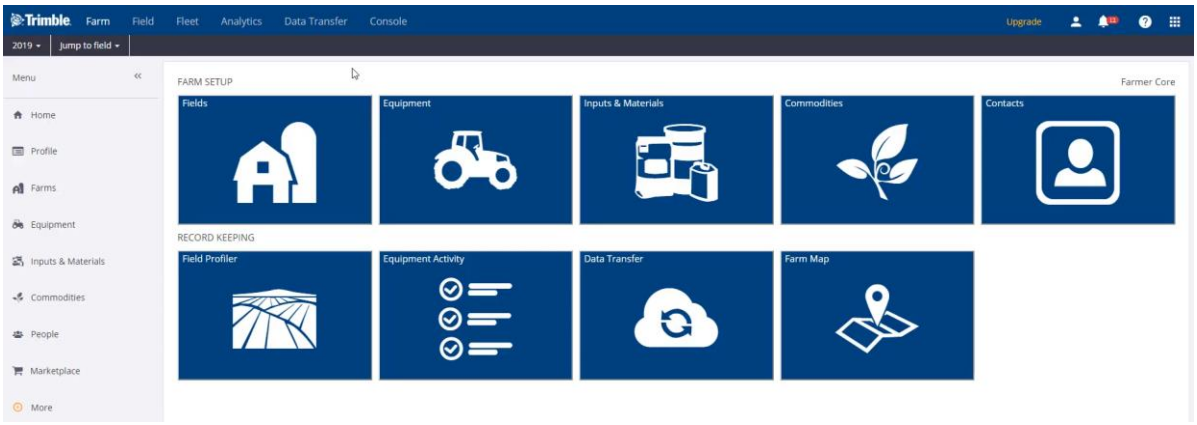

Рисунок 1.4 - Інтерфейс головного меню веб-версії Trimble

Trimble Ag підходить як для малих сімейних ферм, так і для великих аграрних підприємств. Завдяки масштабованості та гнучкості, ця система може бути налаштована відповідно до потреб конкретного господарства, забезпечуючи ефективне управління та максимальну продуктивність.

Останнім аналогом буде ПЗ компанії Agrivi, яка позиціонується як ключовий гравець в управлінні фермерськими господарствами, обслуговуючи не лише фермерів, а й агропродовольчі компанії, банки та кооперативи.

Agrivi — це програмне забезпечення для управління фермерським господарством, яке допомагає аграріям оптимізувати процеси на фермі, підвищувати ефективність і приймати обґрунтовані рішення [3]. Воно призначене для використання в рослинництві, незалежно від розмірів господарства, та пропонує широкий спектр функцій як і попереднє програмне забезпечення.

Agrivi часто порівнюють із такими платформами, як FarmLogs та Trimble Ag Software, але його відрізняє ширший функціонал і акцент на повну цифровізацію фермерських процесів без обов’язкової прив’язки до певного обладнання.

У той час як багато конкурентів зосереджуються на конкретних аспектах, таких як управління технікою або виробництво зерна, Agrivi надає інструменти для управління різноманітними культурами та охоплює весь життєвий цикл виробництва та бізнесу.

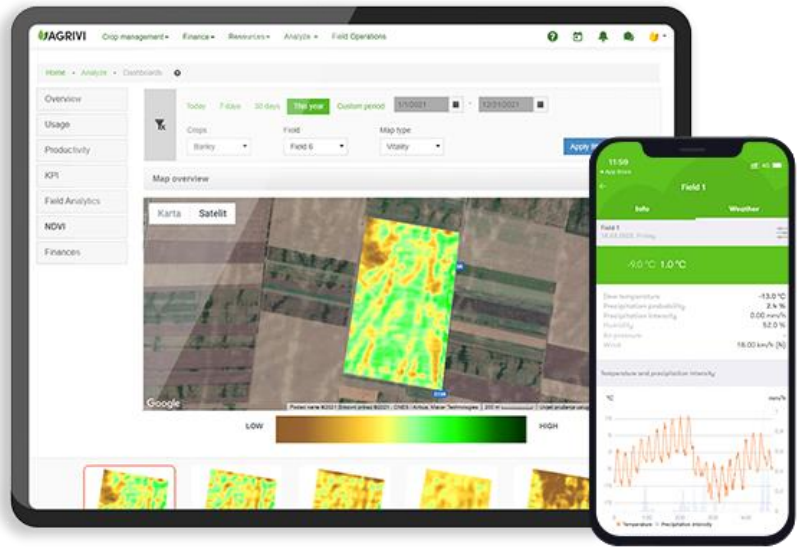

Рисунок 1.5 - Інтерфейс мобільного застосунку Agrivi

Агропланувальник у програмі Agrivi не потребує спеціального обладнання для базового використання. Проте можливе підключення датчиків, дронів чи іншого обладнання для покращення точності даних та автоматизації збору інформації. Також є можливість інтеграції з ERP-системами, такими як SAP або Microsoft Dynamics, для повної оптимізації управління господарством.

Агропланувальник у програмі Agrivi дозволяє створювати завдання для працівників, планувати використання техніки та ресурсів, а також встановлювати часові рамки для кожного етапу робіт. Завдяки модулю Work Orders, ви можете відслідковувати виконання завдань у реальному часі, аналізувати ефективність працівників і оптимізувати процеси. Додаткові функції включають облік витрат праці та механізації, що спрощує управління бюджетом.

Agrivi — це потужне і глобальне програмне забезпечення для управління фермерським господарством, яке забезпечує багатофункціональність і простоту використання. Його можливості нічим не поступаються конкурентам, а в

аспектах, як-от інтеграція з іншими системами та аналіз продуктивності, навіть перевершують їх. Агропланувальник від Agrivi є одним із ключових модулів, що дозволяє створювати завдання для працівників, планувати техніку, керувати ресурсами та аналізувати витрати, забезпечуючи ефективне управління господарством.

Програмне забезпечення, яке ми розглянули, є високоефективним та комплексним. Воно пропонує широкий спектр функцій, включаючи управління технікою, моніторинг стану культур, аналіз врожайності та інтеграцію з різними системами. Тим не менш, ці програми можуть бути досить складними для освоєння, вимагаючи значного часу та зусиль для повного розуміння всього функціоналу та його ефективного використання. Це може створити певні труднощі для фермерів, особливо тих, хто не має попереднього досвіду роботи з подібним програмним забезпеченням.

Окрім цього, варто зазначити, що для власників невеликих господарств такі комплексні рішення можуть виявитися надмірними. На початкових етапах розвитку фермерського бізнесу інвестування в таке програмне забезпечення може бути недоцільним через його вартість та складність. Малі господарства можуть не потребувати всього спектру функцій, що надаються цими програмами, та можуть знайти їх занадто складними для своїх потреб.

Таким чином, хоча John Deere Operations Center, Trimble Ag та Agrivi є потужними інструментами для великих фермерських господарств, вони можуть не підходити для малих бізнесів на початкових етапах. Відсутність десктопних версій також може бути недоліком для деяких користувачів. Це підводить нас до аргументації на користь розробки менш комплексного, але більш доступного та вузько спеціалізованого програмного забезпечення для десктопних платформ, яке буде спрямоване на задоволення потреб невеликих фермерських господарств.

1.3 Постановка задачі створення десктопного додатку для планування агро-робіт

Великі та комплексні аграрні системи, такі як John Deere Operations Center, Trimble Ag та FarmLogs, беззаперечно покривають більшість потреб бізнесу. Але також варто розуміти, що вибір, самостійне освоєння даних програм чи навчання персоналу займає певний час. Для прикладу людині, яка має низькі навички володіння персональним комп'ютером, буде складно розібратися у схожих системах та цілком можливо знадобиться додатковий персонал.

Також виникає питання, чи справді весь функціонал потрібен невеликому підприємству або людині, яка має кілька ділянок для вирощування культур для власного споживання. Далеко не кожному фермеру потрібні, наприклад, ГІС-системи, не всі застосовують спеціальні датчики у своєму господарстві і не вся техніка є передовою та оснащеною модулями для взаємодії з відповідним ПЗ. Але без грамотного планування робіт підприємство може понести значні втрати.

Головна суть всіх цих рішень полягає в ефективному плануванні дій із використанням усіх наявних даних. Мінімум, що потрібен кожному фермеру — це розподіл обов’язків між членами бригади, аналіз того, яку культуру варто посадити, де і які добрива внести тощо.

Проаналізувавши ринок актуального програмного забезпечення для агробізнесу, можна дійти до висновку, що сектор десктопних застосунків значно просідає у порівнянні з мобільними додатками та веб-додатками. Однак це не означає, що рішення призначені для виконання на персональних комп’ютерах не знайдуть своїх користувачів.

Для підтвердження наведемо кілька ключових тез в підтримку десктопного автономного ПЗ:

- Продуктивність та швидкість. Десктопні додатки зазвичай працюють швидше та ефективніше, оскільки вони можуть безпосередньо взаємодіяти з апаратним забезпеченням комп'ютера і не залежать від швидкості інтернет-з'єднання.

- Безпека та конфіденційність. Десктопні додатки часто забезпечують вищий рівень безпеки та конфіденційності, оскільки дані можуть зберігатися локально, а не передаватися через мережу.
- Офлайн-доступ. Десктопні додатки можуть функціонувати без доступу до інтернету, що робить їх більш надійними в умовах обмеженого або нестабільного інтернет-з'єднання.

Якщо створити вдало спроектовану програмну архітектуру, то вдасться розробити ПЗ, яке добре масштабується. Це дозволить спочатку створити базове рішення для продажу цільовим покупцям, а при потребі модернізувати його або додати новий функціонал для задоволення потреб окремих клієнтів. У результаті ми отримуємо дистрибутив, який у найкращому сценарії не потребує постійних доопрацювань, характерних для стрімко розвиваючихся веб-технологій, а також не вимагає витрат на хостинг. Таким чином, ПЗ повністю належить користувачу, і він вільний використовувати його на власний розсуд.

Підсумувавши вищесказане можна сформувати мету нашого проекту - це створення десктопного додатку для планування аграрних робіт, спеціально орієнтованого на потреби малих фермерських господарств.

Наш додаток буде зосереджений на ключових функціях планування робіт, таких як:

- Внесення інформації про техніку, поля та працівників.
- Створення та управління планами робіт.
- Відстеження виконання запланованих робіт.

Цей фокус на основних потребах дозволить забезпечити ефективне управління аграрними процесами без зайвих складнощів.

Розробка десктопного планувальника аграрних робіт для ПК дозволить малим та середнім господарствам ефективно планувати та управляти своєю діяльністю без зайвих витрат та складнощів. Завдяки простоті використання та зосередженню на основних функціях, наш додаток стане надійним інструментом для щоденної роботи, а вибір десктоп платформ забезпечить максимальну доступність та сумісність.

РОЗДІЛ 2. ІНСТРУМЕНТИ ДЛЯ СТВОРЕННЯ ДЕСКТОПНОГО ДОДАТКУ В СИСТЕМІ WINDOWS

2.1 Розгляд поняття десктоп-застосунка

Десктопна програма або настільний додаток - це програма, призначена для запуску на персональному комп'ютері, що характеризується використання ресурсів системи для виконання своїх функцій [20].

Зазвичай такі додатки виконують складні обчислення і потребують прямого доступу до потужностей системи. На відміну від них, веб-додатки зазвичай призначені для простіших завдань і мають менше навантаження на ресурси свого серверу. Наприклад, ПЗ для 3D моделювання потребує значних системних ресурсів. Створення і підтримка великої кількості серверів для таких обчислень можуть бути дуже дорогими. Набагато дешевше купити ПК з оптимальним апаратним забезпеченням і виконувати такі типи робіт безпосередньо на ньому.

Як зазначалося раніше, десктопні програми запускаються локально на пристрої користувача, тоді як веб-програми завантажуються та виконуються через інтернет з віддаленого сервера. Це важливий аспект при порівнянні десктопних та веб-програм.

Кожен варіант має свої переваги і може бути оптимальним залежно від потреб бізнесу. Поки інтернет не стане всепроникним, а настільні комп’ютери не перетворяться на тонкі термінали, що з'єднують користувачів з їхніми цифровими середовищами у глобальній хмарі, десктопні, мобільні та онлайн-додатки продовжуватимуть співіснувати.

Щоб краще зрозуміти суть десктопних додатків, розглянемо всі їх відмінності з веб-застосунками відповідно до табл. 2.1.

Таблиця 2.1 – Порівняння десктоп- та веб-застосунків

Десктоп - застосунки	**Веб - застосунки**
● 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 застосунок	● Застосунок доступний без додаткового завантаження
-	● Працює на усіх платформах, де доступний Браузер

Як ми бачимо на кожен плюс одного з типу цих додатків існує свій мінус. Багато десктоп додатків проектують таким чином, що вони не працюють без доступу до інтернет мережі, хоча це не завжди так. Завдяки тому, що більшість операцій виконується локально, легше забезпечити безпеку даних, встановивши все необхідне ПЗ для захисту системи від шкідливих програм. В свою чергу прослідкувати за цілісністю всіх процесів у веб-застосунку набагато важче. Також варто врахувати, що веб досить вразливий до різного роду кібер-атак, що може вплинути на доступність сервісу для користувачів. Суттєвим плюсом веб-додатків є можливість запуску у браузері з будь-якого пристрою без додаткового завантаження та залучення значних системних ресурсів. Наразі це ключовий фактор домінації веб-застосунків над десктопом.

Також варто розуміти, що існують різновиди десктопного ПЗ. Ось деякі з них [18]:

- Автономні додатки - це типові додатки, що є повністю автономними і не потребують підключення до інтернет мережі.
- Клієнт-серверна програма - це програма, яка працює на комп'ютері, але отримує доступ до інформації з віддаленого сервера.
- Утиліти та плагіни - це все, що може допомогти комп'ютеру або браузеру працювати ефективніше, потрапляє в цю категорію.
- Системні програми та служби - це все, що дозволяє комп'ютеру запускати різні інші програми, потрапляє в цю категорію.
- Мультимедійні програми- це програми, які відтворюють подкасти, фільми, відео, музику та інше.

Загалом існує досить багато різновидів таких застосунків. Кожен з них виконує свої унікальні функції та задачі. Багатьма з них ми користуємося кожен день і не можемо уявити свою роботу без них. Це веб-браузери, програмне забезпечення для редагування фото та відео, різноманітні редактори програмного коду, офісні пакети програм Microsoft.

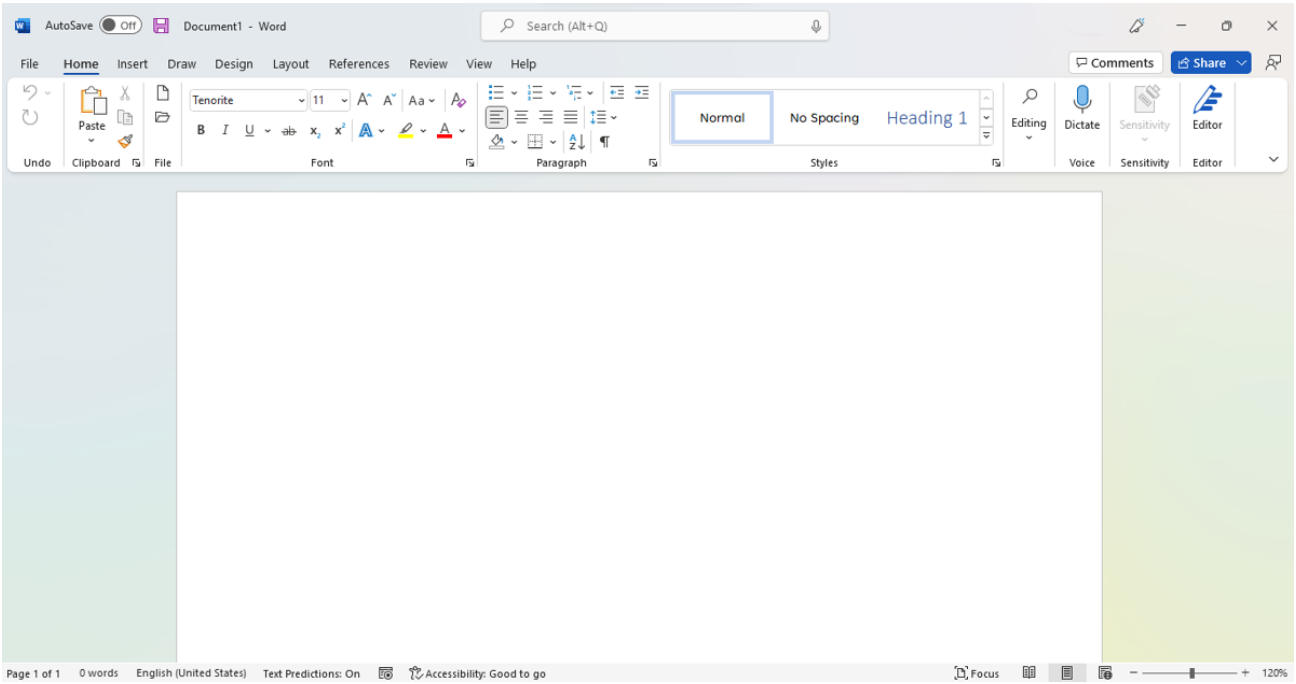

Рисунок 2.1 Інтерфейс десктопної версії Microsoft Word

Десктопні додатки працюють під управлінням різних операційних систем, таких як macOS від Apple, Linux на основі відкритого вихідного коду, а

також Microsoft Windows, яка є однією з найпоширеніших операційних систем у світі.

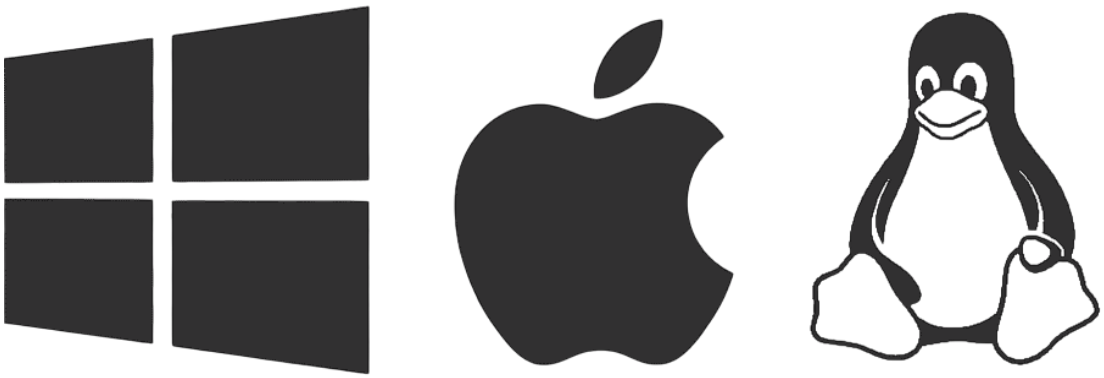

Рисунок 2.2 - Логотипи популярних ОС

Якщо вірити статистиці з вікіпедії, то на 2024 рік у стаціонарних комп'ютерах і ноутбуках найбільше використовується Microsoft Windows - 72,22%, за нею йде Apple macOS - 14,73%, настільна Linux - 3,88% і Google ChromeOS - 2,45%. Оскільки ChromeOS є ОС на базі Linux, її можна додати до загальної частки десктопного Linux, що становить 6,33%. 0,01% - це FreeBSD, а решта 6,7% - це, ймовірно, маловідомі дистрибутиви Linux [28].

Загалом немає статистики щодо ОС які найчастіше використовуються в сільському господарстві і точно лідера визначити складно. Спираючись на статистику порталу Wikipedia та інших ресурсів, більшість бізнесів використовують саме ОС Windows. Таким чином можна дійти до висновку, що основна маса людей використовує ОС Windows для своїх повсякденних та рабочих задач.

2.2 Особливості платформ та фреймворків для розробки десктопних додатків

В першу чергу розглянемо лідируючу платформу ОС Windows та її інструменти для створення нативних додатків.

Windows є однією з найпопулярніших операційних систем у світі завдяки своїй доступності та широкому розповсюдженню. Вона має простий,

інтуїтивний інтерфейс, який підходить для користувачів з різним рівнем досвіду, і використовується як у професійній діяльності, так і для розваг. Windows надає користувачам доступ до величезної кількості інструментів для розробки, що дозволяє створювати нативні додатки для найрізноманітніших завдань.

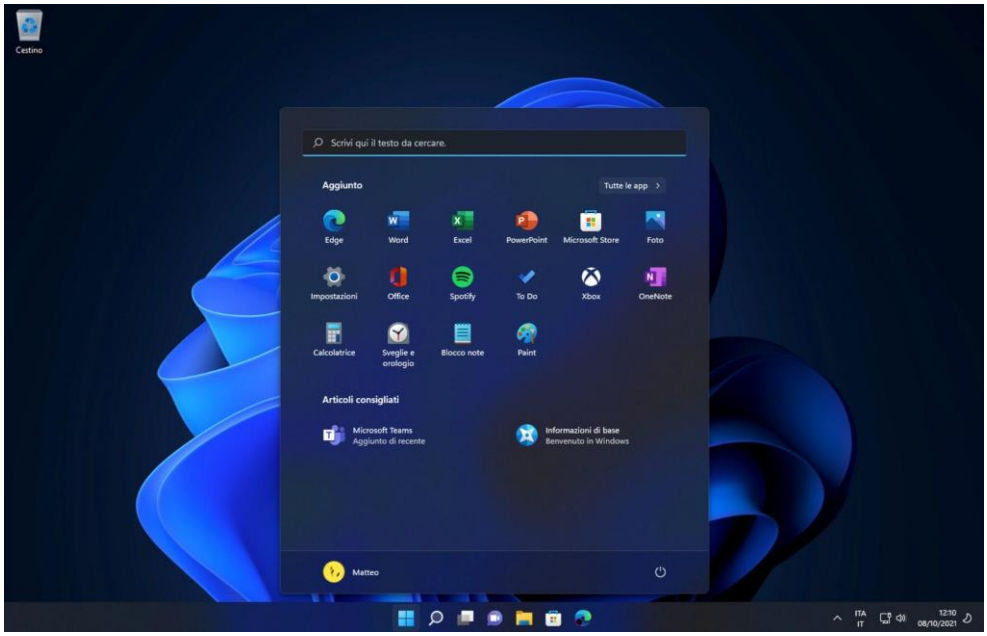

Рисунок 2.3 Інтерфейс Windows 11

Завдяки своїй популярності, Windows також пропонує безліч навчальних матеріалів, форумів і спільнот, що робить її особливо привабливою для новачків. Це сприяє легкому освоєнню платформи та розширює її застосування в різних сферах, забезпечуючи стабільну підтримку і постійний розвиток екосистеми Windows.

Говорячи про нативну розробку для Windows перше, що спадає на думку - це платформа .NET. Дана технологія являє собою платформу з відкритим вихідним кодом для створення настільних, мобільних і веб-додатків, які можуть працювати в будь-якій операційній системі. .NET включає в себе інструменти, бібліотеки та мови програмування, що підтримують сучасну, масштабовану і високопродуктивну розробку програмного забезпечення [27].

Підсумувавши вищесказане можна виділити три основні компоненти .NET:

- Мови програмування .NET. Сюди відноситься C#, F#, Visual Basic та мови спільної мовної інфраструктури по типу Eiffel, IronPython, PowerBuilder і багато інших.
- Платформи моделей додатків. Це різноманітні інструменти та бібліотеки призначені для розробки різних типів додатків. Це інтернет-додатки, мобільні застосунки, додатки для ПК.
- Середовище виконання .NET.

Розробники використовують відповідні мови програмування і платформи моделей додатків для створення ПЗ, а середовище .NET виконує та запускає їх.

Наразі існує три ключових і сучасних фреймворки для створення десктопних додатків за допомогою .NET.

WinForms — одна з перших технологій для створення десктопних додатків у Windows. Вона базується на простій моделі перетягування компонентів у візуальному редакторі, що дозволяє швидко створювати інтерфейси користувача [16]. WinForms забезпечує зручне середовище для розробників-початківців і підходить для побудови додатків малих та середніх розмірів.

WPF (Windows Presentation Foundation) — більш сучасний фреймворк, що використовує XAML для створення гнучких і візуально насичених інтерфейсів. Він підтримує 3D-графіку, анімації, масштабування і прив’язку даних, що дозволяє створювати адаптивні та високопродуктивні додатки для Windows.

.NET MAUI (Multi-platform App UI) — останній із .NET фреймворків, що дозволяє створювати кросплатформні додатки для Windows, macOS, Android та iOS з єдиної кодової бази. Ця технологія, що прийшла на заміну Xamarin.Forms, надає можливості розробки під різні ОС, що робить її перспективним вибором для кросплатформних рішень.

Звичайно кожен з цих фреймворків має свої переваги та використовується в залежності від потреб та масштабів проекту. Однак ці інструменти дозволяють створювати програми будь-якої складності.

Хоча більшість користувачів обирають Windows як основну операційну систему, macOS також має мільйони відданих користувачів. Особливо

популярна вона серед висококваліфікованих фахівців у галузях технічного профілю, дизайну, обробки аудіо та відео.

Серед важливих особливостей macOS є те, що її не можна офіційно встановити на звичайний ПК: для роботи з macOS необхідно купувати комп’ютер Apple. Техніка цієї марки зазвичай дорожча у порівнянні з аналогами, однак розробка програм для macOS можлива лише на цих пристроях. Це обмеження вкотре доводить закритість екосистеми Apple, яка має вищий вхідний бар'єр, але також і високу лояльність користувачів.

Рисунок 2.4 Інтерфейс MacOS Mojave

Програмний фреймворк Cocoa слугує нативним об'єктно-орієнтованим інтерфейсом прикладного програмування для MacOS. Застосунки, розроблені з використанням Cocoa, можуть повністю інтегруватися з системними можливостями macOS, забезпечуючи високий рівень продуктивності та користувацького досвіду.

Cocoa побудовано на основі багаторівневої архітектури операційної системи macOS. За своєю суттю, Cocoa складається з двох основних фреймворків: Foundation та AppKit.

Foundation є базовою частиною Cocoa, що забезпечує основні типи даних, колекції та сервіси операційної системи. Він включає класи і протоколи для керування рядками, числами, датами і колекціями, такими як масиви і словники. Фреймворк також обробляє низькорівневі функції, такі як багатопотоковість, файловий ввід/вивід та мережевий зв'язок.

AppKit побудований на основі фреймворку Foundation. Ця технологія надає компоненти користувацького інтерфейсу та елементи керування, необхідні для додатків macOS. Він включає класи для вікон, меню, кнопок, текстових полів та інших елементів інтерфейсу. AppKit також займається обробкою подій, малюванням та рендерингом користувацького інтерфейсу.

Також якщо говорити про розробку на Mac не можна не згадати про SwiftUI. Це сучасний фреймворк, представлений Apple у 2019 році, який ознаменував амбіції компанії переосмислити розробку інтерфейсів. Завдяки декларативному синтаксису створення користувацьких інтерфейсів стає простішим і доступнішим. Замість традиційного підходу, де інтерфейс визначається крок за кроком, у SwiftUI розробники описують, як інтерфейс повинен виглядати і поводитись в різних станах, а фреймворк автоматично відтворює ці стани на екрані. Тісна інтеграція зі Swift забезпечує сучасний робочий процес розробки, що робить SwiftUI привабливим для тих, хто прагне долучитися до великої екосистеми Apple.

Порівнюючи SwiftUI з Cocoa та AppKit, можна виділити декілька ключових відмінностей. Cocoa є зрілим фреймворком з великою кількістю бібліотек, широкими можливостями кастомізації та інтеграції з існуючими системами. Завдяки багаторічному досвіду та оптимізації, Cocoa часто пропонує кращу продуктивність, особливо у складних проектах. AppKit, як частина Cocoa, є стабільним фреймворком, який гарантує передбачувану роботу в усіх версіях macOS. Він має дещо більші накладні витрати, ніж SwiftUI, але забезпечує надійність і стабільність, необхідні для багатьох додатків.

Завдяки інтуітивним інструментам та декларативному підході SwiftUI ідеально підходить для швидкого прототипування та розробки інтерфейсів

користувача. Він забезпечує швидкий рендеринг, що робить його привабливим вибором для створення динамічних інтерфейсів. Важливо зазначити, що SwiftUI та AppKit можна комбінувати в одному проекті. Це дозволяє використовувати переваги обох фреймворків: стабільність AppKit та простоту SwiftUI.

Вибір між Cocoa, SwiftUI або AppKit залежить не лише від їхніх індивідуальних переваг, але й від конкретних потреб проекту. У той час як Cocoa обіцяє глибину, SwiftUI пропонує швидку розробку для початківців. AppKit, хоч і не є лідером, але залишається надійним вибором для певних застарілих проектів.

Насправді компанія Apple надає хороші інструменти для створення функціональних та візуально-привабливих застосунків, однак багато користувачів відштовхує закритість екосистеми та вартість цієї техніки.

Завершує наш огляд операційна система Linux — безкоштовна і відкрита ОС, яка відома своєю гнучкістю і можливістю повного контролю над середовищем. В основі Linux лежить ядро з відкритим кодом, яке надає користувачам та розробникам свободу налаштовувати систему відповідно до індивідуальних потреб. Завдяки відкритості коду програмісти можуть без обмежень адаптувати Linux під конкретні завдання або навіть створювати власні версії системи.

Ядро Linux є базовим компонентом для численних дистрибутивів, кожен з яких — своєрідна збірка програм і інструментів, що формує завершену операційну систему. Дистрибутиви розробляються спільнотами або компаніями і відрізняються один від одного за інтерфейсом, набором програм та призначенням. Серед найвідоміших — Ubuntu, Fedora, Debian і GNOME, кожен із яких підходить для різного рівня досвідчеості користувачів і задач, забезпечуючи підтримку, зручні інтерфейси й адаптацію під потреби кожного [10].

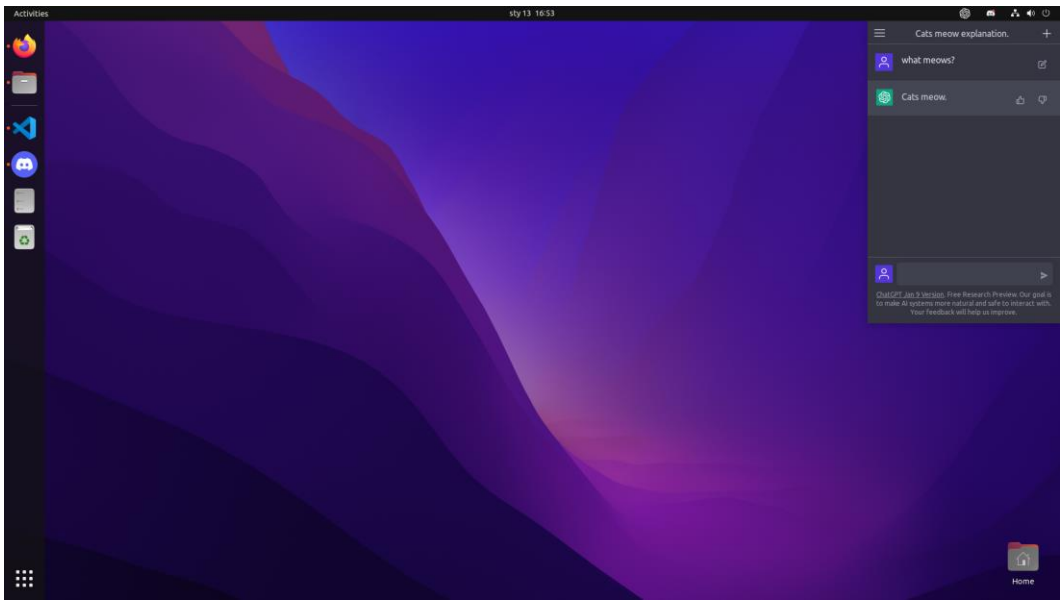

Рисунок 2.5 Інтерфейс робочого середовища GNOME

На відміну від інших ОС Linux фактично немає нативних інструментів для створення десктопних додатків. Однак є досить багато крос платформних фреймворків, тому розглянемо два з найпопулярніших.

Перше, що приходить на думку - це бібліотека GTK. Фактично являє собою кросплатформовий набір інструментів для створення графічних інтерфейсів користувача. Дана технологія активно використовується в операційній системі Linux. Розроблена спершу для графічного редактора GIMP, GTK стала основою для створення застосунків, орієнтованих на середовище GNOME та багато інших Linux-програм. Вона написана на мові C, але має прив’язки до різних мов програмування, таких як Python, C++, Vala, Ruby та інші, що робить її універсальним інструментом для розробників.

GTK надає великий набір графічних елементів, які підтримують сучасний, інтерактивний інтерфейс: кнопки, текстові поля, списки, вкладки, діалогові вікна тощо. Це дозволяє створювати застосунки зі складними інтерфейсами, що чудово інтегруються у середовище Linux. Однією з головних переваг GTK є її модульність і можливість налаштування, що дозволяє змінювати вигляд додатків за допомогою CSS-подібних стилів, адаптуючи їх під сучасні вимоги [11].

Серед відомих програм, що використовують GTK, є такі як GIMP, GNOME Files, Inkscape і навіть платформи як Flatpak. GTK є кросплатформною бібліотекою і, окрім Linux, підтримує Windows і macOS, хоча основним середовищем залишається Linux. Завдяки своїй відкритій ліцензії LGPL GTK активно підтримується спільнотою і GNOME Foundation, а також є вибором багатьох розробників для створення відкритого та нативного програмного забезпечення для Linux.

Окремо варто також розказати про QT. Це потужний кросплатформний фреймворк для створення графічних і консольних застосунків, який широко використовується в середовищі Linux, а також на Windows, macOS і навіть мобільних платформах, таких як Android та iOS. Написаний на C++, Qt надає високопродуктивні засоби для розробки застосунків, а також має прив’язки до Python (PyQt або PySide), що робить його доступним для розробників із різним технічним досвідом. Qt відомий своєю гнучкістю та продуктивністю і є основою для KDE — одного з провідних графічних середовищ Linux [21].

QT надає розробникам великий набір віджетів, потужні інструменти для роботи з 3D-графікою, базами даних, мультимедіа, мережею, а також засоби для створення сучасних інтерфейсів з анімацією та адаптивним дизайном. За допомогою Qt Designer розробники можуть створювати інтерфейси візуально, що спрощує процес проектування. Крім того, Qt підтримує декларативний підхід до створення інтерфейсів через QML, мову, що поєднує JavaScript-подібний синтаксис і можливість декларативно описувати елементи UI.

Завдяки потужності, кросплатформенності та стабільності, Qt є вибором багатьох компаній для створення як комерційних, так і відкритих програм. Відомі програми на основі Qt включають Telegram, VirtualBox, VLC, а також численні програми KDE.

Незважаючи на вільний доступ та деякі переваги, Linux є досить складним у вивченні та налаштуванні для початківців. Його найчастіше використовують у вузькоспеціалізованих технічних завданнях, де цінуються стабільність і можливості тонкого налаштування. Попри це, Linux пропонує надзвичайну

гнучкість для досвідчених користувачів, особливо тих, хто потребує надійного середовища для серверів, розробки або наукових обчислень.

Проаналізувавши доступні платформи, можна зробити висновок, що Windows є найоптимальнішою ОС для нашого застосунку. Створення програм для macOS потребує спеціалізованого обладнання від Apple, що може бути витратним, а також знання вузькоспеціалізованих технологій. Linux, хоч і пропонує високу гнучкість, вимагає глибшого технічного досвіду й має обмежені навчальні матеріали для настільної розробки. Враховуючи це, а також ймовірність того, що наша цільова аудиторія використовує Windows, вибір цієї ОС забезпечує доступність, великий обсяг ресурсів для навчання та нижчий поріг входження.

2.3 Вибір технології для створення десктопного Windows застосунка

Microsoft є лідером у розробці інструментів і технологій для створення десктопних додатків, і фактично монополістом у цій сфері. Завдяки широкому спектру інструментів у межах екосистеми .NET, розробники можуть обирати оптимальний варіант для своїх проектів, залежно від складності та вимог до інтерфейсу.

Ми розглянемо три найактуальніші інструменти для розробки десктопних додатків від Microsoft — Windows Forms, Windows Presentation Foundation та .NET MAUI. Кожен із них має свої переваги і підходить для різних типів завдань, що дозволяє обрати оптимальний підхід для розробки сучасного та функціонального програмного забезпечення.

Першою на огляді буде Windows Forms. Це одна з найперших технологій для створення графічних інтерфейсів на платформі .NET, яка зберігає популярність завдяки простоті використання та великій кількості навчальних матеріалів. WinForms надає розробникам доступ до зручного конструктора інтерфейсів у Visual Studio, який дозволяє створювати форми за допомогою

перетягування та налаштування елементів. Це значно полегшує процес створення інтерфейсу, знижуючи поріг входження для новачків і дозволяючи швидко зібрати робочу програму з базовими функціями.

Windows Forms використовує графічну бібліотеку GDI+ (Graphics Device Interface Plus) для відображення інтерфейсу користувача. Це API, створений Microsoft, який дозволяє малювати прості графічні елементи, такі як кнопки, текстові поля, панелі та інші стандартні UI-компоненти. GDI+ обробляє базові графічні операції, такі як рендеринг форм і тексту, маніпуляція з кольорами та обробка зображень.

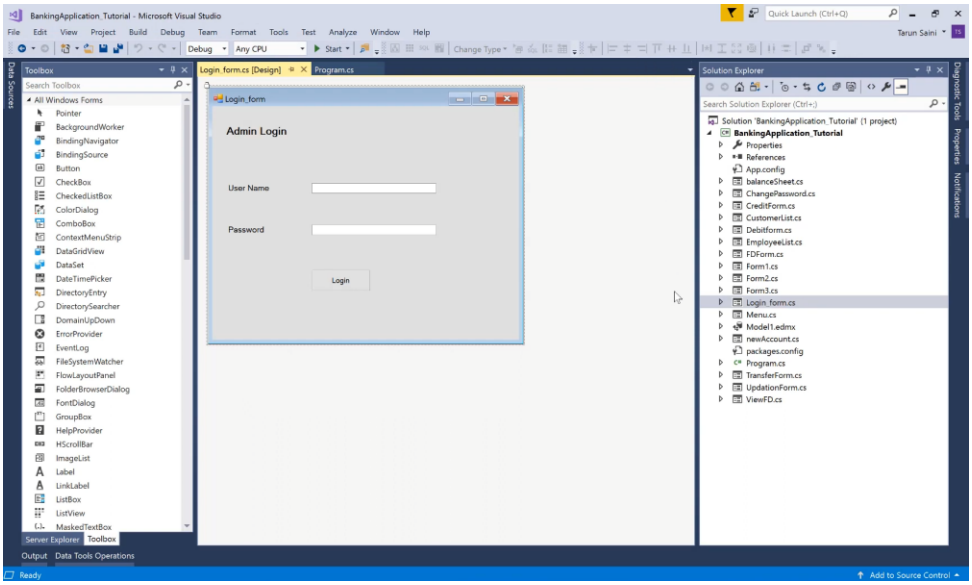

Рисунок 2.6 - Інтерфейс роботи з Windows Forms

Кожен компонент — кнопка, текстове поле чи інші елементи — створюється як об'єкт, яким легко керувати через код або візуальний конструктор. Однак технологія має певні обмеження, особливо в продуктивності та масштабуванні: WinForms не підтримує складні графічні анімації і має обмеження в адаптивності інтерфейсу для різних розмірів екранів.

Додаток Windows Forms фактично є подієво-орієнтованим додатком. На відміну від пакетних програм, більша частина часу витрачається на очікування від користувача якихось дій, як, наприклад, введення тексту в текстове поле або кліка мишкою на кнопку.

Попри ці обмеження, WinForms залишається надзвичайно легким та

інтуїтивним інструментом, з великою кількістю матеріалів для самостійного навчання, що робить його ідеальним для новачків або невеликих проектів, де простота та швидкість розробки є ключовими перевагами.

Іншим досить популярним інструментом є Windows Presentation Foundation або WPF. Це сучасна технологія для розробки десктопних застосунків у .NET, яка дає змогу створювати інтерфейси з використанням складної графіки, анімацій і тривимірних ефектів. На відміну від Windows Forms, WPF працює на основі XAML — мови розмітки, що використовується для опису структури та зовнішнього вигляду інтерфейсу. Цей підхід забезпечує гнучкість у розділенні логіки додатка та UI, полегшуючи співпрацю між розробниками та дизайнерами.

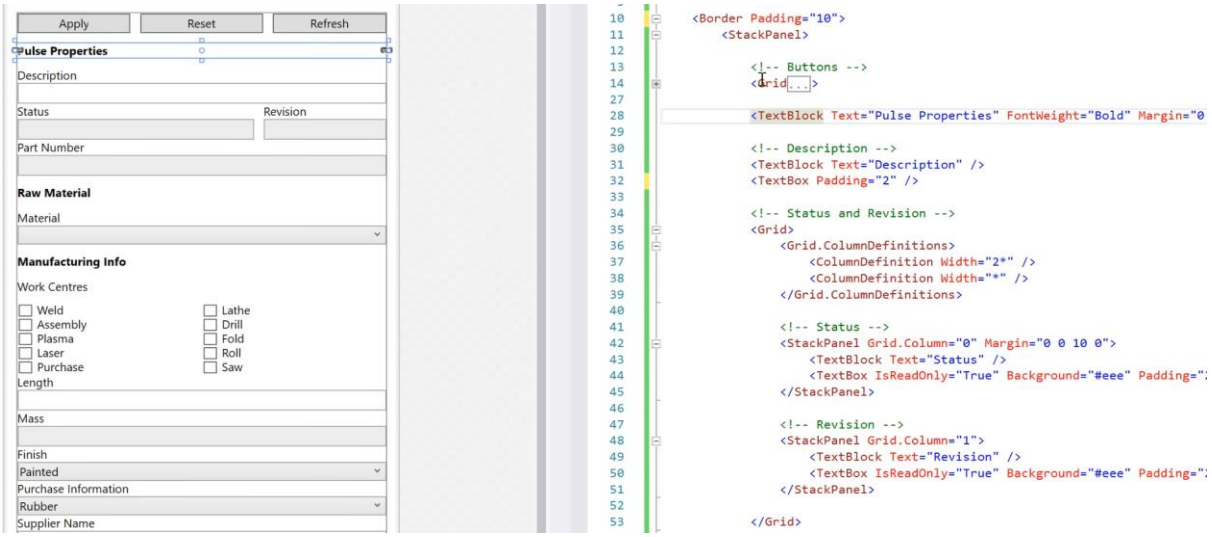

Рисунок 2.7 - Верстка інтерфейсу WPF застосунка

У порівнянні з WinForms, WPF використовує апаратне прискорення та DirectX для рендерингу графіки, що дозволяє реалізовувати складні інтерфейси з високою продуктивністю. На практиці це означає, що WPF працює більш ефективно з мультимедійними елементами, великими обсягами даних і складними анімаціями. Інша важлива перевага — адаптивність: WPF дозволяє легко масштабувати інтерфейс для різних екранів завдяки прив’язці елементів та вбудованій підтримці адаптивного дизайну.

Потрібно розуміти, що навіть при можливості використовувати складніші

та функціональніші елементи інтерфейсу, їх слід ретельно компонувати. Без належного досвіду це може призвести до не оптимізованої програми з нестабільною роботою. XAML забезпечує гнучкі можливості для створення інтерфейсів, проте має досить високий поріг входження, і розробка дизайну вимагатиме більше часу для освоєння та налаштування.

Розроблений Microsoft, .NET MAUI (Multi-platform App UI) - це уніфікований фреймворк для створення крос-платформних додатків для Windows, Android, macOS та iOS. Розробники використовують його для написання коду один раз і розгортання його на різних платформах, усуваючи необхідність дублювати свою роботу в окремій нативній розробці. .NET MAUI є розвитком Xamarin.Forms, успадкувавши його фундаментальні можливості та додавши нові функції та оптимізації.

.NET MAUI для Windows створює застосунок на основі WinUI. Це означає, що інтерфейс вашого застосунку в середовищі Windows буде відповідати стандартам і вигляду застосунків, розроблених на основі Windows SDK. Таким чином, застосунки на .NET MAUI мають уніфікований вигляд і поведінку в межах Windows, забезпечуючи користувачам послідовний досвід, як із рідними застосунками [15].

Варто зазначити, що цей інструмент найбільш корисний, якщо додаток розробляється для Windows з перспективою його запуску на інших платформах. У такому випадку .NET MAUI – чудове рішення. Однак, якщо планів на кросплатформність немає, це може бути не найкращий вибір, і ось чому:

- Нестандартний XAML: Синтаксис XAML у .NET MAUI має відмінності від XAML в WPF або Windows SDK, що потребує додаткового часу для адаптації.
- Обмежені елементи керування: Наразі .NET MAUI не підтримує Windows SDK, тому доведеться шукати заміну стороннім елементам.
- Більша складність: При переході до Windows додатку виконуються додаткові перетворення, що може призвести до проблем зі стилями, API або макетом.

.NET MAUI — це нова, динамічна та перспективна технологія, що дозволяє створювати сучасні кросплатформені застосунки .Завдяки своїй універсальності, .NET MAUI відкриває перед розробниками широкі можливості для розвитку та масштабування додатків на різних платформах. Хоча технологія ще активно розвивається та має обмежену кількість матеріалів, вона обіцяє стати провідним рішенням для мультиплатформеної розробки, забезпечуючи стабільну підтримку та зручність у роботі.

З огляду на попереднє, можна зробити висновок, що .NET MAUI не є найкращим варіантом для розробки нашого Windows-застосунку. Ця технологія є корисною для економії ресурсів завдяки можливості запуску програм на кількох платформах, але вона вимагає значного досвіду розробки та розуміння специфіки кожної з платформ. У нашому випадку її використання є недоцільним.

Порівнюючи технології Windows Forms та WPF, однозначно можна сказати, що остання незважаючи на свої явні переваги у нашому випадку буде програвати старішій технології. Windows Forms - це база для розробки класичних Windows застосунків. Завдяки можливості швидко застосовувати готові елементи шляхом перетягування можна ефективно створювати нескладні застосунки, не вдаючись в тонкощі налаштування окремих елементів. Крім того Windows Forms дозволяє зосередитися на безпосередній логіці застосунку без необхідності вивчати архітектурні патерни чи якісь додаткові технології. Відштовхуючись від цих нюансів можна однозначно сказати, що ми зробили найкращий вибір.

У цьому розділі ми проаналізували популярні ОС та технології розробки десктопних додатків для них. ОС Windows є беззаперечним лідером згідно статистики багатьох ресурсів. Цю систему використовують у всіх сферах включаючи персональне користування та бізнес сегмент. З огляду на аналіз доступних технологій для Windows, ми обрали Windows Forms як оптимальне рішення для швидкого та зручного створення прототипу агропланувальника, що відповідає нашим потребам і дозволяє створити простий та ефективний інструмент для користувачі

РОЗДІЛ 3. РОЗРОБКА ДЕСКТОПНОГО ДОДАТКУ НА БАЗІ API WINDOWS FORMS

3.1 Структура інструментарію Windows Forms

Як вже було сказано раніше Windows Forms — це бібліотека для створення GUI у застосунках на платформі .NET. Вона забезпечує розробникам набір інструментів для швидкого створення віконних програм із зручними інтерфейсами.

Windows Forms пропонує набір базових елементів, які дозволяють розробникам створювати функціональні та зручні графічні інтерфейси користувача. Ці елементи включають форми, які виступають основою для вікон додатків, і контроли, що служать інструментами взаємодії з користувачем.

Форма є головним контейнером, який представляє собою вікно програми. Вона слугує основою для розміщення інших елементів інтерфейсу та визначає зовнішній вигляд і поведінку вікна. Форми можуть бути:

- Головні – основне вікно програми, яке відкривається при запуску.
- Модальні – вікна, які блокують доступ до інших форм програми, доки не будуть закриті (наприклад, діалогові вікна).
- Додаткові – вікна, які відкриваються для виконання певних функцій, але не блокують доступ до основного інтерфейсу.

Другим важливим елементом є контроли. Вони підтримують події, які визначають, як елемент реагує на взаємодію користувача, наприклад, натискання кнопки чи зміну тексту в полі. Кожен контрол має власні властивості, які дозволяють налаштувати його вигляд і поведінку. Завдяки цьому Windows Forms надає широкі можливості для створення гнучких і функціональних інтерфейсів.

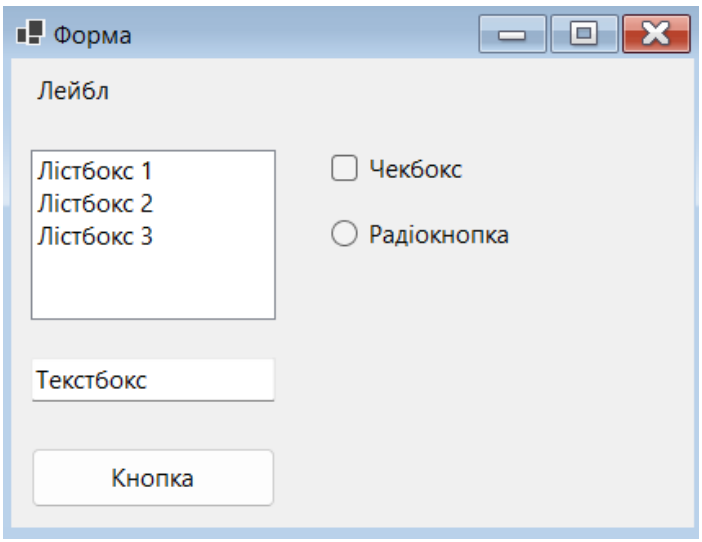

Рисунок 3.1 Основні контроли

Розробники можуть створювати власні контроли, використовуючи базовий клас Control або успадковуючи існуючі елементи, такі як "Button" чи "TextBox", додаючи унікальний функціонал. Для налаштування подій і логіки використовуються делегати та події, наприклад, механізм підписки на події. Крім того, для розширення функціональності можна інтегрувати сторонні бібліотеки, такі як "DevExpress" або "Telerik", які пропонують набір готових розширених контролів і тем оформлення.

Також розробники часто використовують менеджери компоновки. Вони дозволяють автоматизувати розташування контролів на формі, забезпечуючи їх правильне вирівнювання, адаптивність до змін розміру вікна та збереження відступів між елементами. Наприклад, елементи "TableLayoutPanel" (табличне розташування) або "FlowLayoutPanel" (потокове розташування) допомагають створювати інтерфейси, які залишаються зручними та організованими незалежно від розміру вікна.

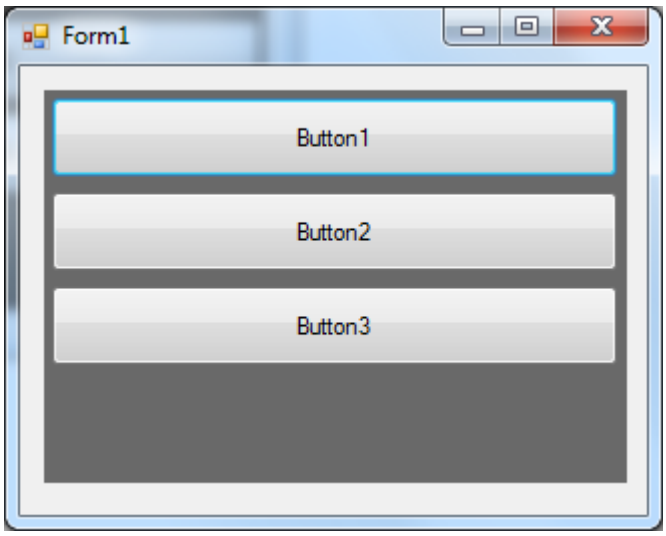

Рисунок 3.2 Вирівнювання кнопок за допомогою "TableLayoutPanel"

Останнім основним елементом інструментарію є події. Вони є ключовим механізмом взаємодії між користувачем і елементами інтерфейсу. Вони дозволяють виконувати певні дії у відповідь на дії користувача або системи. Основні типи подій у Windows Forms включають:

- Події взаємодії з користувачем: наприклад, натискання кнопки , введення тексту , наведення миші.
- Системні події: реагують на зміну стану вікна, наприклад, закриття форми або зміну її розмірів.
- Події клавіатури та миші: такі як натискання клавіші або рух курсора.

Для обробки подій у коді використовується підписка на певну подію, яка викликає відповідний метод. Наприклад, для кнопки можна додати подію "Click", щоб виконати певну логіку після натискання:

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

Рисунок 3.4 Фрагмент коду

Таким чином, менеджери компоновки забезпечують організованість і адаптивність інтерфейсу, тоді як події дозволяють налаштовувати динамічну взаємодію з користувачем.

Visual Studio є основним інструментом для розробки застосунків на основі Windows Forms завдяки своєму зручному інтерфейсу та широкому набору функцій. Це IDE, розроблене компанією Microsoft, яке забезпечує повну інтеграцію з екосистемою .NET. Visual Studio підтримує різні мови програмування, серед яких C#, VB.NET та інші, і пропонує все необхідне для створення, тестування та розгортання додатків [17].

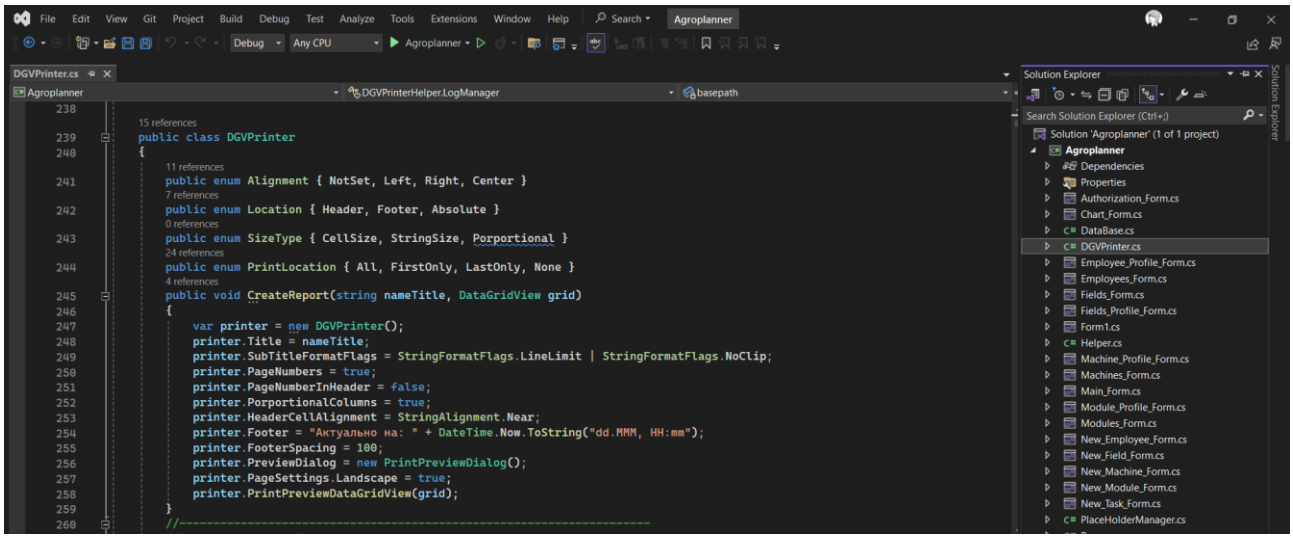

Рисунок 3.5 Робота з кодом у Visual Studio

Інтеграція Windows Forms із Visual Studio надає розробникам потужний візуальний дизайнер, який дозволяє швидко створювати інтерфейси користувача шляхом перетягування елементів із Toolbox. IDE автоматично генерує необхідний код, спрощуючи роботу з формами. Окрім цього, Visual Studio містить вбудовані інструменти для налагодження, аналізу продуктивності та інтеграції сторонніх бібліотек, що робить розробку ефективною та зрозумілою навіть для новачків.

Windows Forms надає простий, але потужний інструментарій для створення настільних застосунків, поєднуючи зручний візуальний дизайн і гнучкість

налаштувань. Завдяки використанню наявного інструментарію розробники можуть легко адаптувати додатки під конкретні потреби, додаючи складні функціональні можливості. Це робить Windows Forms ідеальним вибором для швидкої розробки зручних та ефективних настільних застосунків.

3.2 Створення концепції та проектування десктопного додатку

Концептуальне проектування є підходом до розробки, що акцентує увагу на формуванні чіткої та логічно структурованої концепції системи або продукту [13]. На цьому етапі визначаються основні ідеї, принципи та функції, що лежать в основі майбутнього проекту. Це дозволяє створити загальне бачення та зрозуміти ключові аспекти проекту, які потім будуть деталізовані під час розробки.

Основною метою концептуального проектування є забезпечення цілісного розуміння того, що саме буде створено, і як воно відповідатиме потребам користувачів та вимогам бізнесу. Це допомагає уникнути помилок та непорозумінь на наступних етапах розробки, забезпечуючи злагодженість та послідовність у реалізації проекту. Концептуальне проектування також сприяє ефективному використанню ресурсів та оптимізації процесів розробки, що в кінцевому підсумку підвищує якість кінцевого продукту.

Концептуальне проектування включає кілька ключових етапів, таких як дослідження, аналіз та визначення вимог, формування концепції та прототипування. На етапі дослідження аналізуються потреби користувачів, ринкові тенденції та існуючі рішення. Аналіз та визначення вимог передбачають детальне вивчення функціональних та нефункціональних вимог до системи або продукту. Формування концепції полягає у створенні загальної моделі, яка відображає основні функції та взаємодії системи. Прототипування дозволяє візуалізувати концепцію та перевірити її на практиці, отримуючи ранні відгуки від користувачів та зацікавлених сторін.

Ефективне концептуальне проектування вимагає тісної співпраці між усіма учасниками проекту, включаючи розробників, дизайнерів, аналітиків та користувачів. Це забезпечує врахування всіх аспектів та потреб, що сприяє створенню більш якісного та функціонального продукту. Крім того, концептуальне проектування допомагає виявити потенційні ризики та проблеми на ранніх етапах, що дозволяє вчасно їх вирішувати і запобігати значним затратам на їх виправлення у майбутньому. Таким чином, концептуальне проектування є важливим кроком на шляху до успішної реалізації проекту, що сприяє досягненню його цілей та задоволенню потреб користувачів.

Дослідивши та проаналізувавши ринок аграрного програмного забезпечення ми дійшли до висновку, що існує не так багато аграрних додатків, що виконують одну конкретну функцію. Як правило всі вони є багатофункціональними, а основною їх платформою є веб та мобільні застосунки. Розроблюваний нами додаток, буде розроблений для ключової для всіх бізнесів платформі Windows.

Також перед тим як придумувати базові концепції необхідно врахувати, що підтип нашої десктопної програми - це повністю офлайн застосунок. Тому варто завчасно спланувати яким чином буде відбуватися обмін інформацією і чи потрібен він взагалі.

Розглядаючи функції планування у актуальному програмному забезпеченні для ведення сільськогосподарського бізнесу можна помітити одну ключову схожість. Вона полягає у тому, що в ході планування створюються так звані польові задачі, що містять у собі наступні ключові компоненти:

- Поле на якому виконуються роботи
- Тип робіт
- Техніка для роботи
- Додаткові модулі для техніки
- Дата початку та кінця роботи
- Оператор

Звичайно існують і інші фактори але ці беззаперечно залишаються найважливішими. Розберемо їх детальніше.

Для роботи на тому чи іншому полі важливо розуміти яка культура була посаджена зараз або раніше, чи є поле зараз оброблене і на якій стадії підготовки воно знаходиться. Врахування цих аспектів дозволяє ефективно планувати та оптимізувати сільськогосподарські операції, мінімізуючи ризики та підвищуючи врожайність. Це допомагає приймати обґрунтовані рішення щодо наступних кроків, таких як вибір оптимального часу для посіву або збору врожаю [2].

Типи робіт на полі можна розділити на кілька основних категорій, серед яких обробка ґрунту займає особливе місце. Обробка ґрунту включає безліч різних операцій, таких як оранка, культивація, боронування, глибоке рихлення, дискування та мульчування. Ці заходи спрямовані на поліпшення структури ґрунту, знищення бур'янів та підготовку поля до посіву.

Окрім обробки ґрунту, існують інші важливі типи робіт:

- Посів: Висадка насіння у підготовлений ґрунт із урахуванням оптимальної глибини та відстані між рядами.
- Боротьба з шкідниками: Використання пестицидів та інших засобів для захисту рослин від комах, гризунів та хвороб.
- Полив: Забезпечення рослин водою шляхом зрошення або крапельного поливу, що є критично важливим для підтримання оптимального рівня вологості ґрунту.
- Внесення добрив: Збагачення ґрунту необхідними макро- та мікроелементами для стимулювання росту рослин та підвищення врожайності.
- Збирання врожаю: Завершальний етап, що включає збір і транспортування врожаю з поля до місць зберігання чи переробки.

Кожен з цих типів робіт відіграє важливу роль у забезпеченні ефективного функціонування сільськогосподарського бізнесу та досягненні високих результатів у вирощуванні культур.

Якщо спростити, то існує два основних типи техніки для виконання польових робіт: трактори та комбайни. Комбайни призначені для збирання

врожаю і можуть бути спеціалізовані для різних видів культур, таких як зернові, кукурудза або соняшник. Вони оснащені механізмами для зрізання, обмолоту та очищення зерна, що дозволяє ефективно та швидко зібрати врожай.

Трактори, з іншого боку, виконують всі інші види польових робіт за допомогою спеціальних модулів та навісного обладнання. До таких модулів належать плуги для оранки, культиватори для обробки ґрунту, сівалки для посіву, обприскувачі для внесення пестицидів, розкидачі добрив, косарки та інші інструменти. Завдяки різноманіттю модулів трактори можуть адаптуватися до різних задач і забезпечувати гнучкість та ефективність у сільськогосподарських операціях.

Таким чином, трактори і комбайни є основними інструментами в арсеналі сільськогосподарських підприємств, забезпечуючи виконання всіх необхідних робіт від підготовки ґрунту до збирання врожаю.

Що стосується оператора, то важливо враховувати, чи має він відповідну категорію для управління обома видами транспорту: тракторами та комбайнами. Це дозволяє забезпечити гнучкість у розподілі завдань та ефективне використання ресурсів. Крім того, при плануванні необхідно враховувати зайнятість працівника в інших роботах у певний період, щоб уникнути конфліктів та простоїв у графіку.

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.

Рисунок 3.6 - Ключові аспекти планувальника

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.

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.

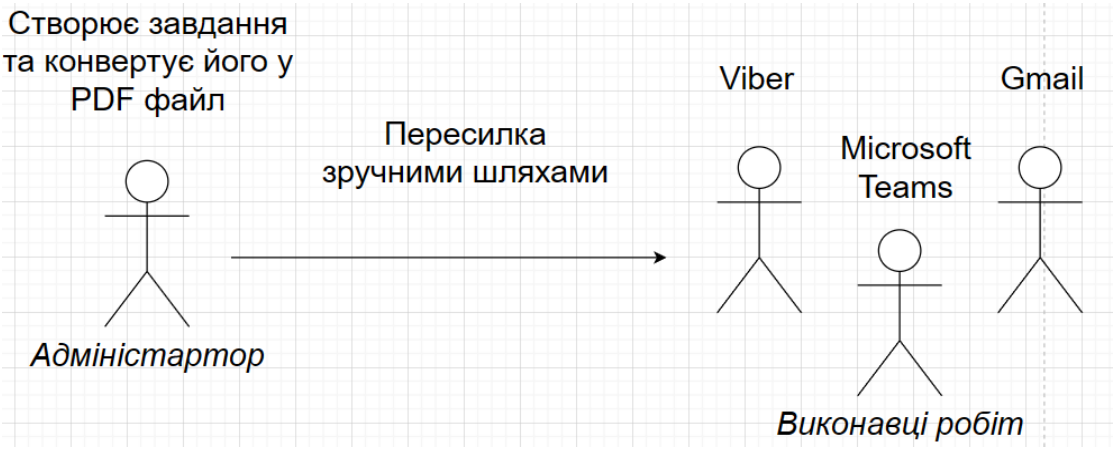

Рисунок 3.7 - Сценарій поширення файлів

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".

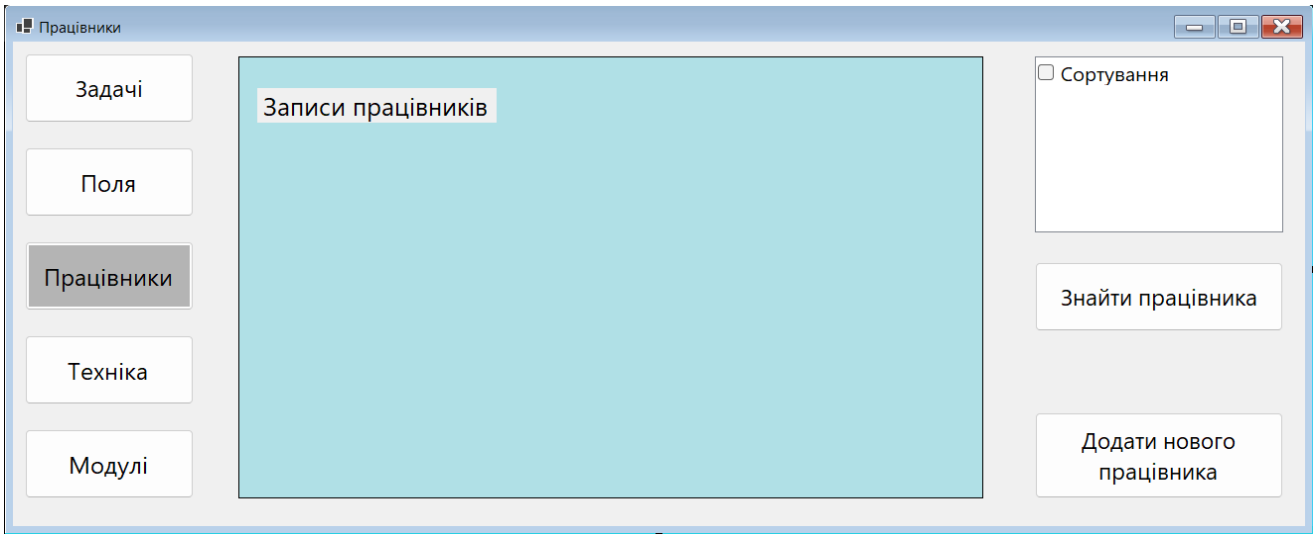

Рисунок 3.8 - Прототип вкладки "Працівники"

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

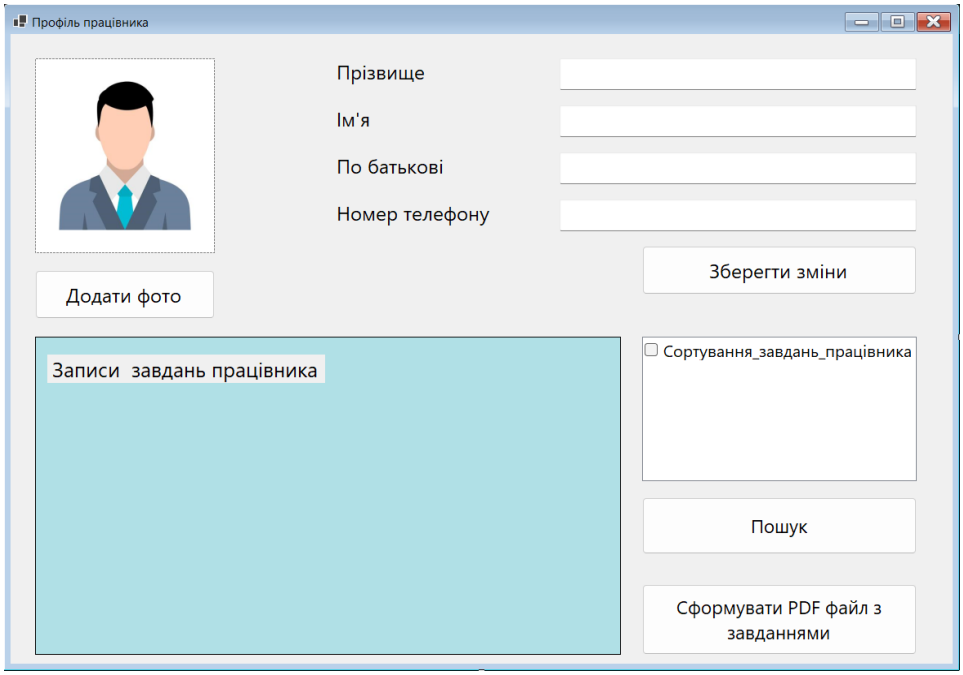

Рисунок 3.9 - Прототип профільної сторінки окремого працівника

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 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 3.10 - 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.

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 3.11 - Таблиця "tasks"

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. 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 додатка А. рис. А.1 ми спостерігаємо 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 знадобляться нам для відображення гістограм.

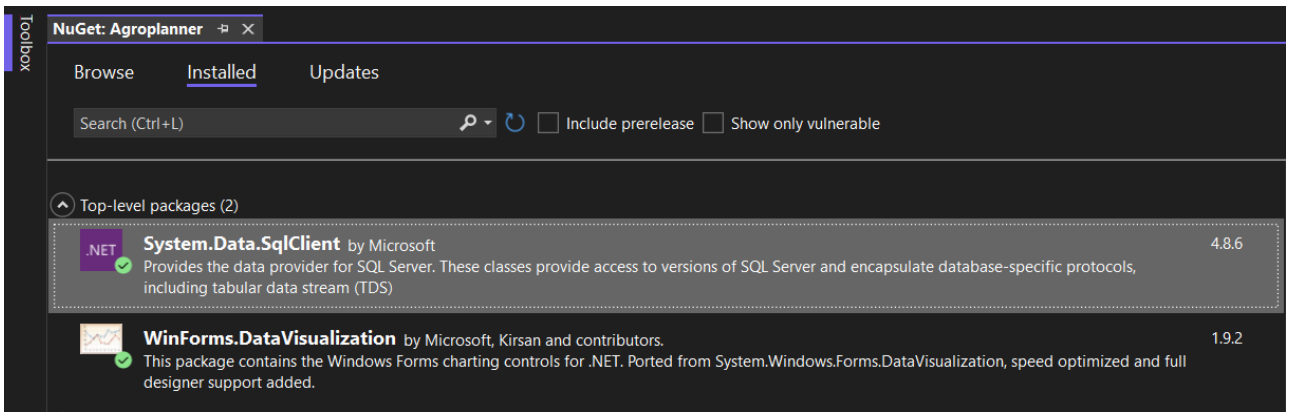

Рисунок 3.12 - Встановлені компоненти

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

```
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"

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.

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.

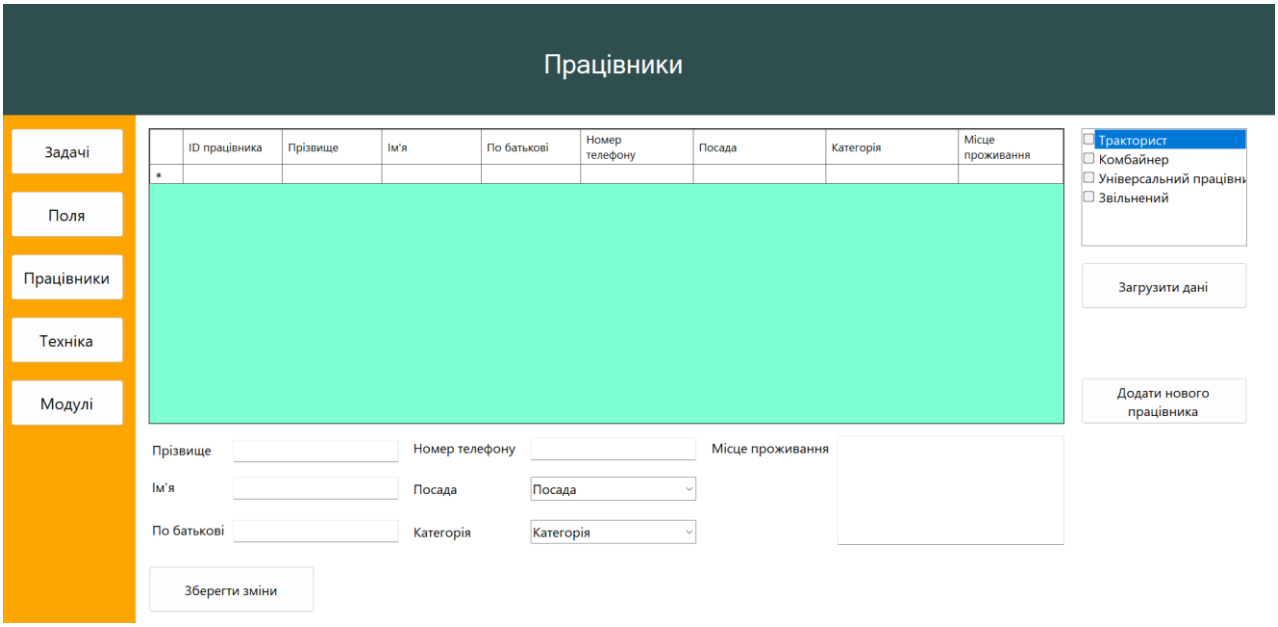

Рисунок 3.14 - Форма "Працівники"

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 застосунків, 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 - Відображення записів таблиці "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 описаний на рисунку дод. Б. рис. Б.1.

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.

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.

Кожен із цих атрибутів відіграє важливу роль у формуванні загального враження користувача про якість продукту. Відповідність цим характеристикам забезпечує не лише задоволеність користувачів, але й підвищує конкурентоспроможність програмного забезпечення на ринку.

Методологія тестування – це система підходів, принципів та методів, які використовуються для перевірки програмного забезпечення з метою оцінки його якості. Вона визначає, як саме проводити тестування, які інструменти

застосовувати та які критерії використовувати для оцінювання результатів. 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.

Також існують різноманітні рівні тестування, які застосовуються на різних етапах розробки програмного забезпечення та часто комбінуються для досягнення кращого результату. Це може бути перевірка окремих компонентів чи модулів, щоб переконатися в їхній коректній роботі, або ж тестування взаємодії між компонентами для оцінки їхньої сумісності. Важливим є також загальне тестування системи, що дозволяє перевірити її функціональність у повному обсязі, а також тестування відповідності бізнес-вимогам. Додатково використовуються підходи, що враховують доступ до коду чи його відсутність, залежно від специфіки завдань.

Тестування програмного забезпечення є невід’ємною складовою процесу розробки, яка забезпечує його якість, надійність і відповідність вимогам користувачів. Вибір правильної методології, поєднання ручного та автоматизованого тестування, а також застосування різних рівнів перевірки дозволяють ефективно виявляти помилки на всіх етапах життєвого циклу ПЗ.

Завдяки тестуванню розробники знижують ризики, підвищують конкурентоспроможність продукту та забезпечують кінцевим користувачам позитивний досвід роботи із системою.

4.2. Тестування десктопного застосунку

Для нашого навчального проекту ми провели тестування методом "чорного ящика". Цей підхід фокусується на перевірці роботи застосунку 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

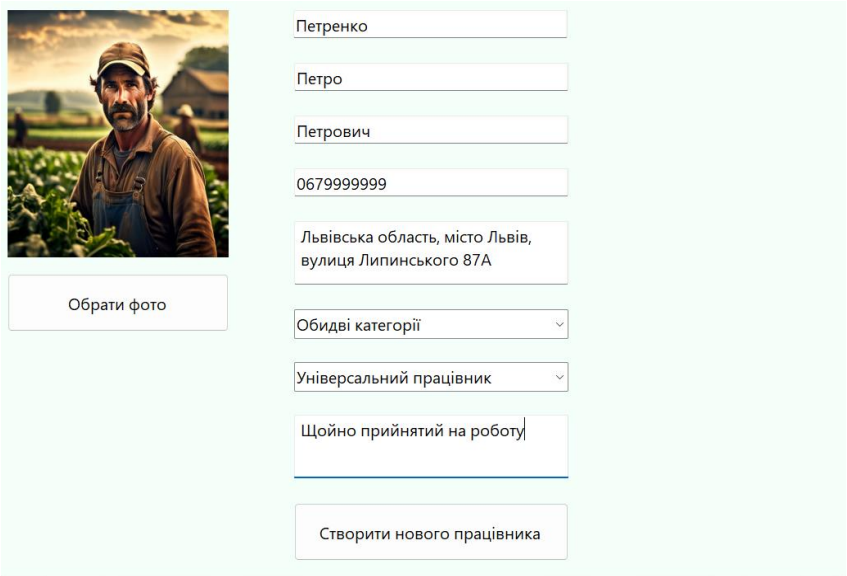

Рисунок 4.1 - Форма створення нового працівника

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.

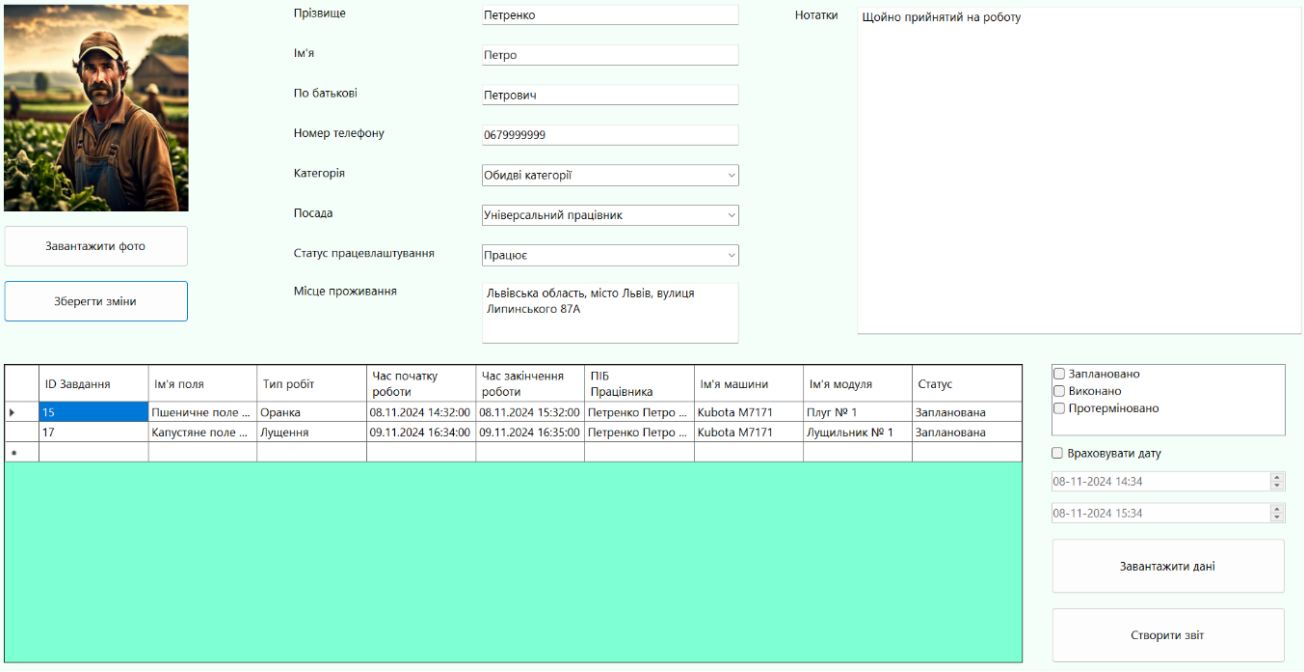

Рисунок 4.2 - Профіль працівника

Як ми бачимо на рисунку 4.2 уся інформація збереглася 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 працівника у PDF-форматі

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 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 застосунка: 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. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [1]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній із них присвоїмо ймовірність виникнення:

Таблиця 5.1 – Ймовірність виникнення небезпечних подій

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0,2+ 0,4 – 0,2 \cdot 0,4 = 0,0592$.

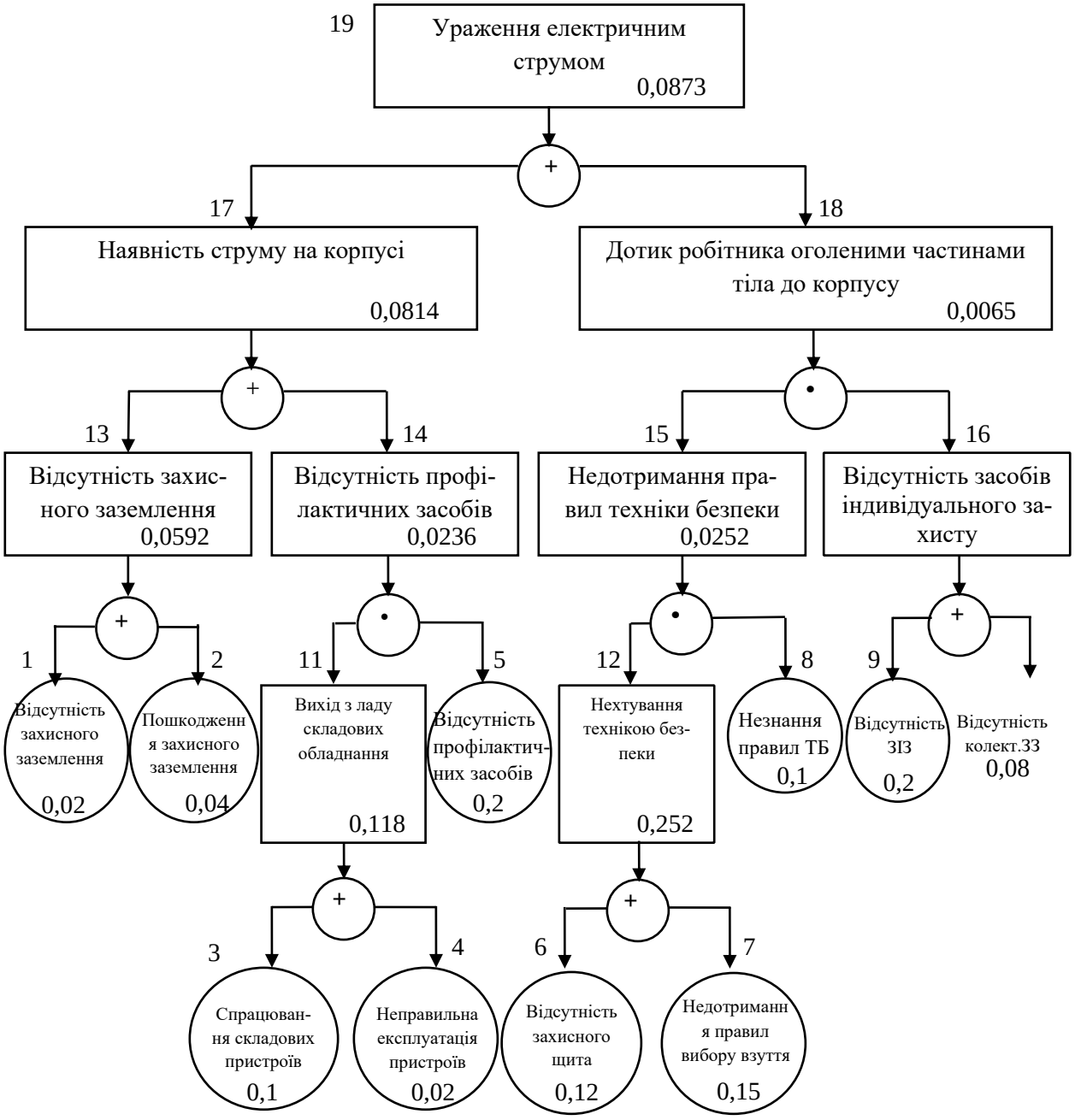

Рис. 5.1. Матриця логічних взаємозв’язків між окремими подіями травмонебезпечної ситуації [1]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 – P_4P_5 = 0,3 + 0,4 – 0,3 \cdot 0,4 = 0,118.$$

$$P_{12} = P_6 + P_7 – P_6P_7 = 0,3 + 0,5 – 0,3 \cdot 0,5 = 0,252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0,2 + 0,15 - 0,2 \cdot 0,15 = 0,264.$$

$$P_{14} = P_{11} \cdot P_5 = 0,118 \cdot 0,2 = 0,0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0,252 \cdot 0,1 = 0,0252.$$

$$P_{17} = P_{13} + P_{14} - P_{13} \cdot P_{14} = 0,592 + 0,0236 - 0,0592 \cdot 0,0236 = 0,0814.$$

$$P_{18} = P_{15} \cdot P_{16} = 0,264 \cdot 0,0252 = 0,0065.$$

$$P_{19} = P_{17} + P_{18} - P_{17} \cdot P_{18} = 0,0065 + 0,0814 - 0,0065 \cdot 0,0814 = 0,0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращення санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- престиж та задоволення працею.

5.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином [1]:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

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 застосунок для зручного та інтуїтивного агропланування.

У наступному розділі ми дослідили концепцію настільного застосунка, 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 застосунків. 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 застосунку, 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: 2009 р. 264 с.

2. Осадчий О. Основи Сільського господарства. Навчальний посібник: 2021 р., 294 с.

3. Agrivi. URL: https://www.agrivi.com/products/360-farm-enterprise/ (дата звернення 04.09.2024).

4. Apple Documentation. URL: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html (дата звернення 28.08.2024).

5. EOS data analytics. URL: https://eos.com/blog/gis-in-agriculture/ (дата звернення 26.08.2024).

6. Existek. URL: https://existek.com/blog/best-frameworks-for-desktop-application-development/ (дата звернення 29.08.2024).

7. Folio3 AgTech. URL: https://agtech.folio3.com/blogs/best-crop-management-software-solutions/ (дата звернення 28.08.2024).

8. Fortune business insights. URL: https://www.fortunebusinessinsights.com/farm-management-software-market-110388 (дата звернення 25.08.2024).

9. GlobalX. URL: https://globalx-ua.com/internet-veschey-v-selskom-hozyaystve (дата звернення 29.08.2024).

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

11. GTK. URL: https://www.gtk.org/ (дата звернення 27.08.2024).

12. Infopulse. URL: https://www.infopulse.com/blog/software-solutions-agriculture (дата звернення 27.08.2024).

13. Jackson, Daniel. Software Abstractions: Logic, Language, and Analysis. Revised Edition. 2006. 341 p.

14. John Deere. URL: https://operationscenter.deere.com/ (дата звернення 02.09.2024).

15. Learn Microsoft .NET MAUI. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-9.0 (дата звернення 01.10.2024).

16. Learn Microsoft Windows Forms. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-9.0 (дата звернення 02.10.2024).

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

18. Nathan, Adam. WPF 4: Unleashed. 1st Edition. 2011. 848 p.

19. Osherove, Roy. The Art of Unit Testing: With Examples in C#. 2nd Edition. 2013. 296 p.

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

21. QT. URL: https://www.qt.io/product/framework (дата звернення 28.08.2024).

22. Radix. URL: https://radixweb.com/blog/desktop-application-development-guide#Types (дата звернення 25.08.2024).

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

24. SafetyCulture. URL: https://sourcetrace.com/blog/cloud-computing-agriculture/ (дата звернення 28.08.2024).

25. SourceTrace. URL: https://safetyculture.com/topics/smart-farming/ (дата звернення 24.08.2024).

26. Trimble Ag. URL: https://ptxtrimble.com/en/products/software/trimble-agriculture-software (дата звернення 03.09.2024).

27. Troelsen, Andrew. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming. 2022. 632 p.

28. Wikipedia. URL: https://en.wikipedia.org/wiki/Usage_share_of_operating_systems (дата звернення 05.09.2024).

ДОДАТКИ

Додаток А

Зв’язки між таблицями

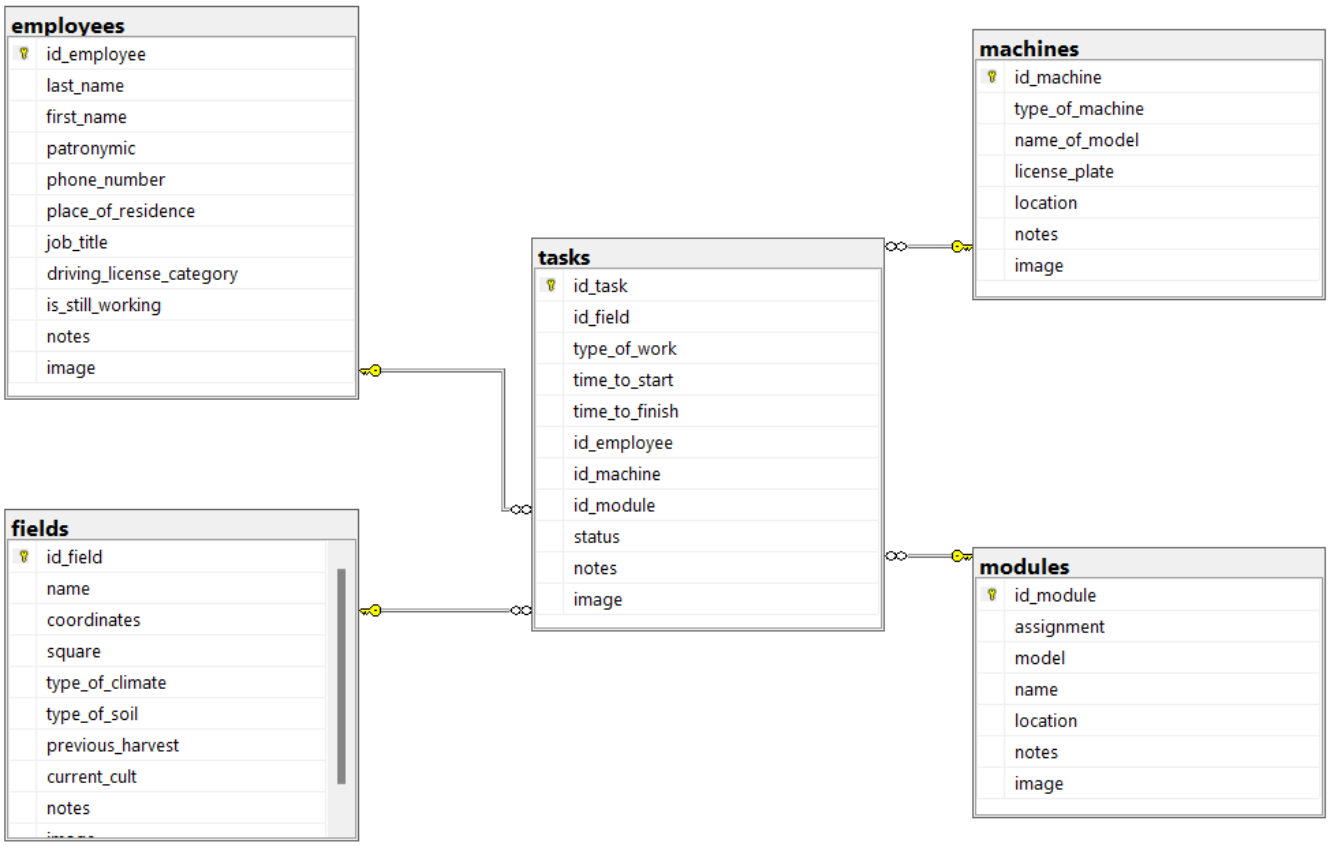

Рисунок А.1 - Зв’язки між таблицями

Додаток Б

Блок-схема форми створення нових завдань

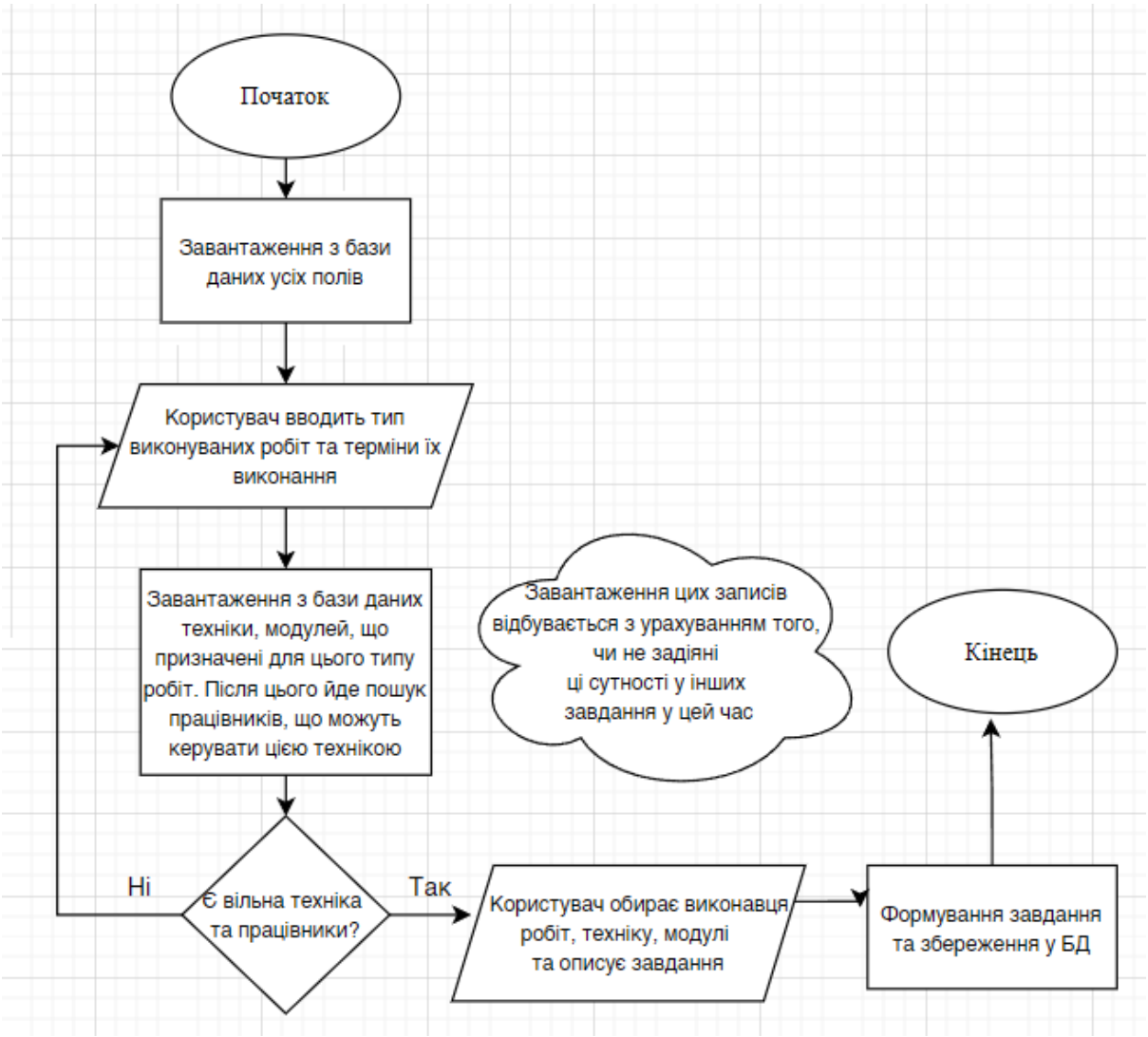

Рисунок Б.1 Блок-схема форми створення нових завдань

Додаток В

Фрагменти коду десктопного застосунку

```
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 В.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 В.2 - Методи форми "Працівники"