ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

Навчально-науковий інститут заочно-дистанційного навчання
Форма навчання заочна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту

Завідувач кафедри

_____Олена ОЛЬХОВСЬКА
(підпис)

«____»__ _____2023 р.

КВАЛІФІКАЦІЙНІ РОБОТА

на тему
«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МАГАЗИНУ ДЕТАЛЕЙ СІЛЬГОСП ТЕХНІКИ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Пархоменко Данило Олександрович

_____ «___»_____2023р.
(підпис)

Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна

_____ «___»_____2023р.
(підпис)

Рецензент _____

ПОЛТАВА 2023 р.

Затверджую
Зав. кафедрою _____
к. ф.-м. н. Олена ОЛЬХОВСЬКА
«__»_____202__ р.

Погоджено
Науковий керівник _____
к. ф.-м. н. Оксана ЧЕРНЕНКО
«__»_____202__ р.

План

кваліфікаційної роботи на тему
«Розробка програмного забезпечення для магазину деталей сільгосп техніки»
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Пархоменко Данило Олександрович

Здобувач вищої освіти _____ Пархоменко Данило

«____» _____ 2023 р.

ЗАТВЕРДЖУЮ

Завідувач кафедри _____Олена ОЛЬХОВСЬКА

«____» вересня 2023р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення для магазину деталей сільгосп техніки»
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Пархоменко Данило Олександрович
Затверджена наказом ректора № ____-Н від «___» _____ 2023 р.
Термін подання студентом роботи «__» _____ 2023 р.
Вихідні дані до кваліфікаційно роботи: публікації з теми, навчальні тренажери в дистанційних курсах із комп'ютерних наук.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП

1. ПОСТАНОВКА ЗАВДАННЯ

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Переваги програмного забезпечення

2.2. Актуальність теми

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Основні поняття програмного забезпечення

3.2. Сучасний процес розробки програмного забезпечення

3.3. Процес розробки програмного забезпечення

3.4. Обґрунтування вибору мови програмування

4. ПРАКТИЧНА ЧАСТИНА

4.1. Основні поняття програмного забезпечення

4.2. Розробка блок-схеми, яка підлягає програмуванню

4.3 Опис роботи програмного забезпечення

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

Перелік графічного матеріалу:1 аркуш блок-схеми, 12 рисунків.
Консультанти розділів кваліфікаційної роботи

Розділ	ПІБ, посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Постанова задачі	к.ф.-м.н., доц. Черненко О.О.		
Інформаційний огляд	к.ф.-м.н., доц. Черненко О.О.		
Теоретична частина	к.ф.-м.н., доц. Черненко О.О.		
Практична реалізація	к.ф.-м.н., доц. Черненко О.О.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій і стандартів та звіт керівнику		
3. Постановка завдання		
4. Інформаційний огляд джерел бібліотек та Інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформленння		
8. Доповідь студента на кафедрі		
9. Доопрацювання (за необхідності), рецензування		

Дата видачі завдання «__» _____ 2023 р.
Здобувач вищої освіти Пархоменко Данило Олександрович
Науковий керівник к. ф.-м. н., Черненко О. О.

Результати захисту кваліфікаційної роботи
Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ЄКТС)

Протокол засідання ЕК № ____ від «____» _____ 2023 р.
Секретар ЕК _____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Записка: 44 стор., в т.ч. основна частина 34 стор., джерел – 10.

Тема роботи – Проектування програмного забезпечення для магазину деталей сільгосп техніки

Мета роботи – проектування програмного забезпечення для магазину деталей сільськогосподарської техніки, полягає у розробці ефективного та функціонального програмного продукту, призначеного для автоматизації процесів управління магазином деталей сільськогосподарської техніки.

Об’єкт роботи – процес прийому та створення нового заказу в сільгосп магазинах.

Предмет роботи – програмне забезпечення для магазинів сільгосп техніки.

Методи дослідження – для розроблення програми було обрано середовище IDE MS Visual Studio з мовою програмування C#.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАГАЗИН СІЛЬГОП ТЕХНІКИ, C#.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовного позначення, скорочень, символів
ДК	Дистанційний курс.
ПЗ	програмне забезпечення
C++	Мова програмування.
C#	Мова програмування.
IDE MS Visual Studio	Середовище для написання програмного коду.

ЗМІСТ

ВСТУП

Програмне забезпечення для магазинів сільгосп техніки є необхідним для ефективного управління продажами, складським обліком, веденням фінансової звітності та управлінням персоналом. Воно дозволяє автоматизувати процеси, збільшити продуктивність та зменшити ризик помилок. Крім того, програмне забезпечення допомагає зберігати дані клієнтів та робити прогнози продажів, що допомагає підвищити прибутковість бізнесу.

Метою роботи є створення програмного забезпечення для магазинів сільгосп техніки.

Об’єктом розробки є процес прийому та створення нового заказу в сільгосп магазинах.

Предметом розробки є програмне забезпечення для магазинів сільгосп техніки.

Перелік використаних методів – застосування математичних основ теорії програмування з теми «Формальні граматики», універсальна платформа для автоматизації сучасних веб-проектів Netlify, мова програмування C#, фреймворк Angular.

Програмне забезпечення готове до використання в магазинах сільгосп техніки.

Робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі описано огляд програмного забезпечення, актуальність теми. В третьому розділі представлено огляд матеріалу за темою роботи, алгоритмізацію задачі за темою роботи, обґрунтування вибору програмних засобів. В четвертому розділі описано розробку блок-схеми, процес програмної реалізації, роботу програмного забезпечення.

Обсяг пояснювальної записки: 44 стор., в т.ч. основна частина – 34 стор., 1 додаток, джерела – 10 назв.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Основним завданням роботи є створення програмного забезпечення для магазин деталей сільгосп техніки.

Розглянемо основні завдання роботи:

1. описати постановку задачі;
2. описати інформаційний огляд;
3. переглянути теоретичний матеріал з теми та навести основні його поняття для використання в програмному забезпеченні;
4. розробити алгоритм програмного забезпечення;
5. скласти блок-схему роботи алгоритму;
6. описати мову програмування та інші технології, що використовувалися при розробці;
7. описати процес реалізації основних етапів створення програмного забезпечення;
8. розробити програмне забезпечення.

Програмне забезпечення повинне бути:

1. зручним;
2. зрозумілим;
3. правильно написаним стосовно частини програмування.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. ПЕРЕВАГИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виділимо основні переваги використання програмного забезпечення для магазину сільгосп техніки:

1. Збільшення ефективності: програмне забезпечення може допомогти автоматизувати процес збирання та зберігання інформації про кожну продукту, тобто зберегти детальний опис продукту, такий як виробник, номер моделі, розмір, матеріали, фотографії і багато іншого. Це дозволить працівникам магазину швидко знайти необхідну інформацію про продукти.

2. Збільшення точності та точності: програмне забезпечення може допомогти зберегти точність та точність даних про продукти. Завдяки цьому, клієнти матимуть доступ до правильної інформації про продукти, що дозволить їм зробити вірний вибір.

3. Покращення обслуговування клієнтів: програмне забезпечення дозволить працівникам магазину швидко знайти товар, який шукає клієнт. Також, детальна інформація про кожну деталь допоможе працівникам надати клієнтам професійні поради за вибіром продукту, що підійде їхній техніці.

4. Збільшення продажів: програмне забезпечення допоможе підвищити продажі шляхом моніторингу продажів продуктів і проведення аналізу. Детальна інформація про продукти допоможе магазину підібрати рекламні та маркетингові акції, що підвищать продажі товарів.

5. Збільшення безпеки даних: програмне забезпечення дозволить захистити інформацію про продукти. Це знизить вірогідність зламу та втрати цінної інформації.

6. Зручність управління: програмне забезпечення допоможе покращити управління магазину через збереження та організацію детальної інформації про продукти. Дані збираються та зберігаються в одному місці, що дозволяє з легкістю відслідковувати та контролювати наявність, продажі та рух продуктів.

2.2 АКТУАЛЬНІСТЬ ТЕМИ

На сьогоднішній день існує відносно велика кількість програмного забезпечення для бізнесу [8]. Але воно має свої недоліки:

1. Висока вартість: існуюче програмне забезпечення може бути дуже дорогим, особливо для невеликих магазинів, що може викликати фінансові труднощі.

2. Складність: багато програмних продуктів мають складний інтерфейс, що робить їх важкодоступними для користувачів з обмеженим досвідом в роботі з комп'ютерами та програмним забезпеченням.

3. Недостатня інтеграція: деякі програмні продукти можуть не підтримувати інтеграцію з іншими програмними продуктами чи іншими технологіями, що використовуються в магазині. Це може призвести до складнощів при обміні даними між програмами.

4. Некоректна робота: деякі програми можуть мати проблеми з функціональністю, таким як помилки в роботі системи, що можуть призвести до збоїв та втрати даних.

5. Потреба у навчанні: необхідно витратити час та гроші на навчання працівників використанню нового програмного забезпечення, що може бути ресурсом витрат.

6. Труднощі зі зміною: перехід на нове програмне забезпечення може знадобитися перетворення даних, що може зайняти дуже багато часу та ресурсів, а деякі клієнти можуть відчути нестабільність в роботі магазину та незручності.

Тому розробка нового програмного забезпечення для магазину сільгосп техніки – дуже важлива та необхідна задача для покращення та спрощення його роботи.

Для розробки програмного забезпечення необхідно ретельно проаналізувати кожен етап створення програми з точки зору її функціональності та відповідності потребам користувачів. Ключовим етапом є визначення вимог до програмного забезпечення, що відбувається на почат-

ковому етапі проєктування. Далі слід здійснити аналіз і планування, щоб визначити технічні можливості для реалізації задач, а також обговорити та узгодити замовлення з клієнтом.

У подальшому процесі розробки необхідно створити базову архітектуру програмного забезпечення та розробити функціональні модулі, які будуть доповнювати і підтримувати роботу програми. При цьому необхідно враховувати технічні та економічні обмеження, а також забезпечувати сталу інтеграцію з іншими системами та пристроями.

Варто зазначити, що розробку програмного забезпечення можна здійснити з використанням різноманітних методів, таких як водоспад, спіральний, гнучкий та інші. У процесі вибору підходу до розробки, слід враховувати розмір та складність проекту, доступність ресурсів та часові обмеження.

Оглядаючи розробку програмного забезпечення на прикладі раніше створеної програми, можна виявити, що успішна розробка залежить від ретельного планування та аналізу, а також відповідності продукту вимогам клієнта та користувачів. Завдяки цьому можна досягти високої якості програмного забезпечення, яке буде відповідати потребам користувачів та буде успішно використовуватись в їх повсякденній роботі.

На наступному етапі розробник виконав інтеграцію створеної програми з існуючими системами магазину, що дозволило забезпечити безперебійну та ефективну роботу програмного забезпечення.

Для забезпечення безпеки та конфіденційності даних, розробник створив механізми авторизації та аутентифікації користувачів, а також забезпечив захист бази даних від несанкціонованого доступу.

Окрім цього, розробник передбачив можливість додавання нових виробників, комплектів, робіт та типових несправностей, що дозволяє розширювати функціональність програми та адаптувати її під конкретні потреби магазину.

У програмі передбачені зручні інструменти для роботи з клієнтами, що дозволяють зберігати та оновлювати інформацію про них, а також створювати відомості замовлень та генерувати гарантійні талони для клієнтів.

Таким чином, створене програмне забезпечення є комплексним інструментом для ефективного управління замовленнями та клієнтами магазину сільськогосподарської техніки, що дозволяє забезпечити швидку та якісну обробку замовлень та підвищити задоволеність клієнтів від обслуговування.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. ОСНОВНІ ПОНЯТТЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення (програмні засоби) – сукупність програм системи оброблення інформації та програмних документів, необхідних для експлуатації цих програм [1]. Це набір інструкцій, які розповідають комп'ютеру, як працювати. Це на відміну від апаратного забезпечення, з якого побудована система і фактично виконує роботу.

Розрізняють:

1. системне програмне забезпечення (зокрема, операційна система, транслятори, редактори, графічний інтерфейс користувача);
2. прикладне програмне забезпечення, що використовується для виконання конкретних завдань, наприклад, статистичне програмне забезпечення;
3. інструментальне програмне забезпечення (комп'ютерні програми, призначені для проєктування, розробки, адміністрування і супроводження системного та прикладного програмного забезпечення).

Виконання програмного забезпечення комп’ютером – це маніпулювання інформацією та керування апаратними компонентами комп’ютера. Наприклад, для персональних комп'ютерів характерно відтворення інформації на екрані та отримання її з клавіатури.

Програмне та апаратне забезпечення є двома взаємодоповнюючими компонентами комп’ютера, і межа між ними нечітка: деякі частини програмного забезпечення на практиці реалізуються суто апаратним забезпеченням комп’ютерних мікросхем, а програмне забезпечення, у свою чергу, здатне виконувати (емулювати) функції електронного обладнання. По суті, призначення програмного забезпечення полягає в тому, щоб контролювати як сам комп’ютер, так і інші програми та маніпулювати інформацією.

Комплекс програм, що забезпечує керування компонентами комп’ютерної системи, такими як процесор, оперативна пам’ять, пристрої

введення-виведення, мережеве обладнання, що виконує роль «міжшарового інтерфейсу», з одного боку якого знаходиться апаратне забезпечення, а з іншого – користувач. програми. На відміну від прикладного програмного забезпечення, системне програмне забезпечення не вирішує конкретних практичних завдань, а лише забезпечує роботу інших програм, надаючи їм сервісні функції, абстрагуючись від деталей апаратно-програмної реалізації комп'ютерної системи, керує апаратними ресурсами комп'ютера. система. Віднесення того чи іншого програмного забезпечення до системного є умовним і залежить від домовленостей, які використовуються в конкретному контексті. Як правило, системне програмне забезпечення включає в себе операційні системи, широкий клас зв'язного програмного забезпечення.

3.2. СУЧАСНИЙ ПРОЦЕС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Інжиніринг програмного забезпечення передбачає отримання результату певної якості, у межах встановленого бюджету та графіка. До списку основних показників якості програмного забезпечення входять зручність супроводу, надійність, ефективність, зручність використання. Процес розробки програмного забезпечення – це структура, перелік активностей, за якою побудовано розробку програмного забезпечення

Існує кілька стандартних процесів розробки програмного забезпечення. В якості прикладу можна взяти водоспадну модель яка включає такі активності:

1. Визначення вимог;
2. Проектування ПЗ та системне проектування;
3. Імплементація та юніт-тестування;
4. Інтеграція та системне тестування;
5. Використання та обслуговування.

Схема інших моделей передбачає виконання перерахованих вище процесів циклічно, в іншому порядку тощо. Перед фахівцями з програмного

забезпечення сьогодні стоять такі проблеми:

1) Проблема успадкування розробленого раніше програмного забезпечення. Чимало великих програмні системи, що експлуатуються в даний час, створені багато років тому, але досі виконують свої функції належним чином. Проблема наслідування означає підтримку та модернізацію таких систем, причому при мінімальних фінансових та тимчасові витрати;

2) Проблема зростання кількості видів програмних систем. В даний час програмне забезпечення має здатне працювати як системи. розподілені в комп'ютерних мережах, що складаються з комп'ютерів різних типів та використовують різні операційні системи. Проблема полягає в тому, що необхідно розробляти надійні програмні системи, здатні працювати спільно з ПЗ різних типів;

3) Проблема зменшення часу створення ПЗ. Багато традиційних технологій створення якісного програмного забезпечення вимагають великих тимчасових витрат. Водночас сьогодні запити ринку ПЗ та вимоги до програмних систем змінюються дуже швидко. Тому і ПЗ має змінюватися з відповідним швидкістю. Проблема, породжена вимогою зменшення часу створення ПЗ, полягає в тому, щоб скоротити час на розробку великих і складних програмних систем без зниження їх якості.

3.3. ПРОЦЕС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Процес розробки програмного забезпечення – сукупність ряду послідовних дій, спрямованих на розробку програмного забезпечення.

Етапи процесу:

Процес розробки складається з багатьох підпроцесів або дисциплін [3]. У моделі водоспаду вони слідують один за одним, в інших подібних процесах змінюється їх порядок або склад.

1. Аналіз вимог → Специфікація ПЗ
2. Дизайн програмного забезпечення
3. Програмування

4. Тестування програмного забезпечення

5. Системна інтеграція

6. Реалізація програмного забезпечення (або встановлення програмного забезпечення)

7. Підтримка програмного забезпечення

Моделі процесів:

1. Водоспад (каскадна, послідовна) модель.

Етапи проекту за каскадною моделлю:

1) посилення вимог;

2) проектування;

3) реалізація;

4) тестування;

5) реалізація;

6) експлуатація та обслуговування.

2. Ітераційна модель

Модель передбачає поділ життєвого циклу проекту на послідовність ітерацій, кожна з яких нагадує «міні-проект», включаючи всі процеси розробки, застосовані для створення менших фрагментів функціональності, порівняно з проектом у цілому. Метою кожної ітерації є створення версії програмної системи, яка працює та включає функціональні можливості, визначені інтегрованим вмістом усіх попередніх і поточних ітерацій. Результат остаточної ітерації містить усі необхідні функції продукту. Таким чином, із завершенням кожної ітерації продукт отримує приріст – приріст – своїх можливостей, які, отже, розвиваються. Ітеративність, інкрементальність і еволюційність у цьому випадку є вираженнями того самого значення в різних словах з дещо різних точок зору.

3. Спіральна модель

Кожна ітерація відповідає створенню фрагмента або версії програмного забезпечення, уточнює цілі та характеристики проекту, оцінює якість отриманих результатів і планує роботу наступної ітерації.

На кожній ітерації оцінюється наступне:

- ризик перевищення термінів і витрат проекту;
- необхідність виконання ще однієї ітерації;
- ступінь повноти і точності розуміння системних вимог;
- доцільність припинення проекту.

Важливо розуміти, що спіральна модель – це не альтернатива еволюційній моделі, а спеціально розроблений варіант. На жаль, спіральна модель часто помилково використовується як синонім еволюційної моделі взагалі, або згадується як цілком самостійна модель. Відмінною рисою спіральної моделі є особлива увага до ризиків, що впливають на організацію життєвого циклу та контрольні точки.

Стадії циклу розробки програмне забезпечення:

1.Аналіз вимог.

Життєвий цикл розробки починається із стадії аналізу, під час якого учасники процесу обговорюють вимоги, що висуваються до кінцевого продукту. Мета цієї стадії – визначення детальних вимог до системи. виключно того, необхідно впевнитися в тому, що всі учасники вірно зрозуміли поставлені задачі і те, як саме кожна вимога буде реалізована на практиці.

Часто в обговоренні беруть участь також і спеціалісти з тестування, які вже на стадії розробки потребують у власність побажання і, при необхідності, коригувати процес.

В залежності від обраної моделі розробки, можуть відрізнятися підходи до визначення моменту переходу з однієї стадії на іншу. Наприклад, у каскадній або V-моделі стадії вимоги аналізу закріплені в документі – вимоги специфікації до програмного забезпечення (Software Requirement Specification, SRS), оформлення якого повинне бути завершене до переходу на наступний етап.

Таким чином, на цьому етапі вимагається збір до програмного забезпечення, що розробляється, їх систематизація, документування, аналіз, а

також виявлення ти рішення протиріччя.

2.Проектування.

На стадії проектування (яка також називається стадією дизайну та архітектури) програмності та системні архітектори, керуючись вимогами, розробляють високорівневий дизайн системи. Різноманітні технічні питання, які розвиваються в процесі проектування, обговорюються зі всіма цікавими сторонами, включаючи замовника. Визначаються технології, які будуть використані в проекті, завантаженість команди, обмеження, часові рамки та бюджет. відповідно з уточненими вимогами обираються максимально відповідні проектні рішення. Затверджений дизайн системи вибирає перелік програмних компонентів, які необхідно розробити, взаємодію з третіми сторонами, функціональні характеристики програми, бази даних, які необхідно використовувати та багато іншого. Дизайн, як правило, фіксується виділеним документом – дизайн-специфікацією (Design Specification Document, DSD). На цьому етапі спрощення процесу візуалізації використовують як такі звані нотації – схематичні системи зображення, що розробляється. Основні нотації:

– Блок-схеми;

– ER-діаграми;

– UML-діаграми;

– Макети – наприклад, намальований у фотошопі прототип сайту.

3.Розробка та програмування.

Після того як вимоги і дизайн продукту затверджені, відбувається перехід до наступної стадії життєвого циклу – деякі розробки. Тут починається написання програмістами коду програми відповідно до визначених раніше вимог. Системні адміністратори покращують програмне оточення, передні програмні засоби розробляють користувальницький інтерфейс програми та логіку її взаємодії з сервером. Крім того, програми пишуть Unit-тести для перевірки правильності роботи коду кожного компонента системи, виконують рев'ю написаного коду, створюють білди і

розгортають готове ПЗ у програмному середовищі. Цикл повторюється до тих пір, поки всі вимоги не будуть реалізовані. Програмування передбачає чотири основні стадії:

1) Розробка алгоритмів – фактично, створення логіки програми роботи;

2) Написання джерела коду;

3) компіляція – перетворення в машинний код;

4) Тестування та відладка – мова йде, головним чином, про юніт-тестування.

4.Документація.

Цей етап виділено достатньо умовно, після того як ми бачили, ті чи інші документи створені на всіх стадіях програми життєвого циклу. Проте, окрім проектної документації та супровідних розробок записів, також і інші текстові документи, що описують, наприклад, функції та способи її використання. Всього існує чотири рівня документації:

1. Архітектурна (проектна) – наприклад, дизайн-специфікація. Це документи, що описують моделі, методологію, інструменти та засоби розробки, обрані для даного проекту.

2. Технічна – вся документація, що супроводжує розробку. Сюди входять різноманітні документи, які пояснюють розробку системи на рівні окремих модулів. Як правило, пишеться у вигляді коментарів до джерела коду, які потім структуруються у вигляді HTML-документів.

3. Користувач – включає довідникові та пояснюючі матеріали, що завершилися кінцевим користувачем для роботи з системою. Це, наприклад, Readme та Userguide, розділ довідки за програмою.

4. Маркетингова – включає рекламні матеріали, які супроводжують випуск продукту. Її мета – у яскравій формі презентувати функціональність та конкурентні переваги продукту.

5.Тестування.

Основні етапи тестування ми вже розглядали раніше, в розділі Фундаментальний процес тестування. Тестувальники займаються пошуком

дефектів у програмному забезпеченні і порівнюють описану у вимогах системи поведінку з реальною. У фазі тестування ви наявні пропущені при розробці баги. При виявленні дефекту, тестувальник складає звіт про помилку, яка передається розробникам. Останні його виправляють, після чого тестування повторюється – але це раз для того, щоб переконатися, що проблема була виправлена, і саме виправлення не стало причиною появи нових дефектів у продукті. Тестування повторується до тих пір, поки не будуть досягнуті критерії його закінчення. Види, методи та техніка тестування ми детально розглянемо в цьому довіднику.

6.Впровадження та супровід.

Коли програма опротестована і в ній більше не залишилося серйозних дефектів, прийшов час релізу і передав її кінцевим користувачам. Після випуску нової версії програми в роботу включається відділ технічної підтримки. Його співробітники забезпечують зворотний зв’язок з користувачами, їх консультування та підтримку. У разі виявлення цих користувачів чи інших пострелізних багів інформація про них передається у звіти про помилки команди розробки, яка, залежно від серйозності проблем, або починає випускати виправлення, або відкладає його до наступної версії програми. Крім того, команда технічної підтримки допомагає зібрати та систематизувати різні метрики – показники роботи програми в реальних умовах.

Отже, всі стадії життєвого циклу ПЗ, які представлені вище, застосовуються в будь-якій моделі розробки, але їх тривалість і порядок проходження можуть відрізнятися.

3.4. ОБҐРУНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET [7]. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

Сучасні технології програмування переживають період бурхливого розвитку. Причинами цього явища є зростання можливостей комп'ютерної техніки, її здешевлення, швидкий розвиток всесвітньої мережі Інтернет, поява все нових і нових сфер застосування комп'ютерів, потреба в різноманітному програмному забезпеченні [9].

Сучасне програмне забезпечення стає все більш складним. Простий у використанні інтерфейс, розширені можливості, управління програмними подіями, нові нетрадиційні сфери застосування – все це призводить до ускладнення програмного забезпечення. Сьогодні створення сучасної програми – це більше, ніж просто зв'язування певних машинних інструкцій, операторів мови високого рівня або навіть наборів процедур і модулів. Основним питанням була розробка виразної структури програми, придатної для легкої модифікації, вільної від помилок і стійкої до змін [10].

C# – об'єктно-орієнтована мова програмування з безпечною системою введення для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класу, атрибути, події, властивості, винятки, коментарі у форматі XML. Запозичивши багато речей у своїх попередників – мов C++, Delphi, Modula і Smalltalk – C#, спираючись на практику їх використання, виключає деякі моделі, які виявилися проблемними при розробці програмних систем: так, C# не підтримує кілька успадкування класів або похідних типів. C# створювався як компонентна мова програмування, і це одна з головних переваг мови, спрямована на можливість повторного використання створених компонентів. Серед інших об'єктивних факторів відзначимо наступне: C# створювався паралельно з Framework Net і повністю враховує всі його можливості – як FCL, так і CLR; C# є повністю об'єктно-орієнтованою мовою, де навіть вбудовані типи мови представлені класами; C# – потужна об'єктно-орієнтована мова з можливостями успадкування та узагальнення; C# є наступником мов C/C++, зберігаючи найкращі риси цих популярних мов програмування; завдяки .Net Framework, який став надбудо-

вою над операційною системою, програмісти на C# отримують ті ж переваги роботи з віртуальною машиною, що й програмісти на Java. Продуктивність коду навіть покращилася, оскільки середовище виконання CLR є компілятором проміжної мови, тоді як віртуальна машина Java є інтерпретатором байт-коду; потужна фреймворкова бібліотека підтримує зручність створення різних типів додатків на C#, дозволяючи легко створювати веб-сервіси, інші типи компонентів, а також просто зберігати та отримувати інформацію з бази даних та інших сховищ даних; реалізація, яка поєднує надійну та ефективну генерацію коду, є важливим фактором, що сприяє успіху C#. Однак C# спочатку був розроблений як мова програмування на рівні додатків для CLR, і тому значною мірою покладається на можливості самого CLR. Це стосується в першу чергу системи типів C#. Наявність або відсутність певних експресивних особливостей мови залежить від того, чи можна певну функцію мови перекласти у відповідні конструкції CLR. Отже, у міру розвитку CLR від версії 1.1 до 2.0 сам C# став набагато багатшим; такої взаємодії слід очікувати в майбутньому. Однак цю схему було порушено з випуском C# 3.0, які є розширеннями мови, які не покладаються на розширення .NET framework. CLR надає C#, як і всім іншим мовам, орієнтованим на .NET, багато функцій, яких бракує «класичним» мовам програмування.

До переваг мови C# можна віднести:

1. Універсальність: мова C# може використовуватися для розробки програмного забезпечення різних типів, таких як настільні та веб-додатки, сервіси та інші.

2. Об'єктно-орієнтований підхід: мова C# підтримує об'єктно-орієнтований підхід, що дозволяє розробляти код, який більш читабельний та структурований.

3. Висока продуктивність: C# компілюється у вказаний мовою код машинний код, що робить його дуже продуктивним для виконання складних задач.

4. Велика підтримка: C# підтримується компанією Microsoft, що забезпечує стабільну та надійну підтримку мови та інших інструментів розробки.

5. Можливість розробки під Windows та інші платформи: C# може використовуватися на різних операційних системах, серед яких Windows, Linux та Mac OS.

Недоліки мови C# включають:

1. Обмеження на платформу: C# розроблена спеціально для платформи .NET, що може бути захищеною від інших операційних систем.

2. Відносна складність: для того, щоб повноцінно засвоїти мову C#, може знадобитися досвід програмування та дуже багато часу на навчання.

3. Відносно повільність: в порівнянні з мовою C++, C# може вимагати більше ресурсів серверу та менше продуктивний.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. ОСНОВНІ ПОНЯТТЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У сучасному світі багато бізнесів стикаються з необхідністю автоматизації своїх процесів. Одним із таких бізнесів є продаж сільгосптехніки. Для ефективного управління замовленнями та обробки інформації про товари, роботи та клієнтів необхідно використовувати спеціальне програмне забезпечення.

Авторизація: перед початком роботи з програмним забезпеченням користувач повинен авторизуватися, введенням свого логіну та пароля.

Налаштування: програмне забезпечення має функцію налаштування, в якій користувач може змінювати параметри програми, такі як мову інтерфейсу, вигляд, розміщення кнопок та інше.

Прийом товару: користувач може додавати новий товар до бази даних, вказавши назву, опис, виробника, модель, серійний номер та інші характеристики. Також можна вказати ціну, кількість одиниць товару та дату поставки.

Прийом замовлень: користувач може створювати нові замовлення, вибираючи необхідні товари зі списку. Він може вказати реквізити замовлення, такі як дату, номер, штрих-код, а також додати клієнта з бази даних, вказавши його ПІБ, телефон та замітку.

Роботи: користувач може додати роботу, яка виконувалась з товаром, вказавши її назву, опис, кількість робочих годин та інші характеристики.

Гарантії: користувач може додати гарантійні талони до замовлень та товарів, вказавши дату початку та закінчення гарантійного періоду.

Список клієнтів: користувач може додавати нових клієнтів до бази даних, вказуючи їх ПІБ, телефон та адресу.

Типові несправності: користувач може створювати типові несправності, які можуть виникати з товарами, вказуючи їх назву та опис.

Статус замовлення: корист.

4.2. РОЗРОБКА БЛОК-СХЕМИ, ЯКА ПІДЛЯГАЄ ПРОГРАМУВАННЮ

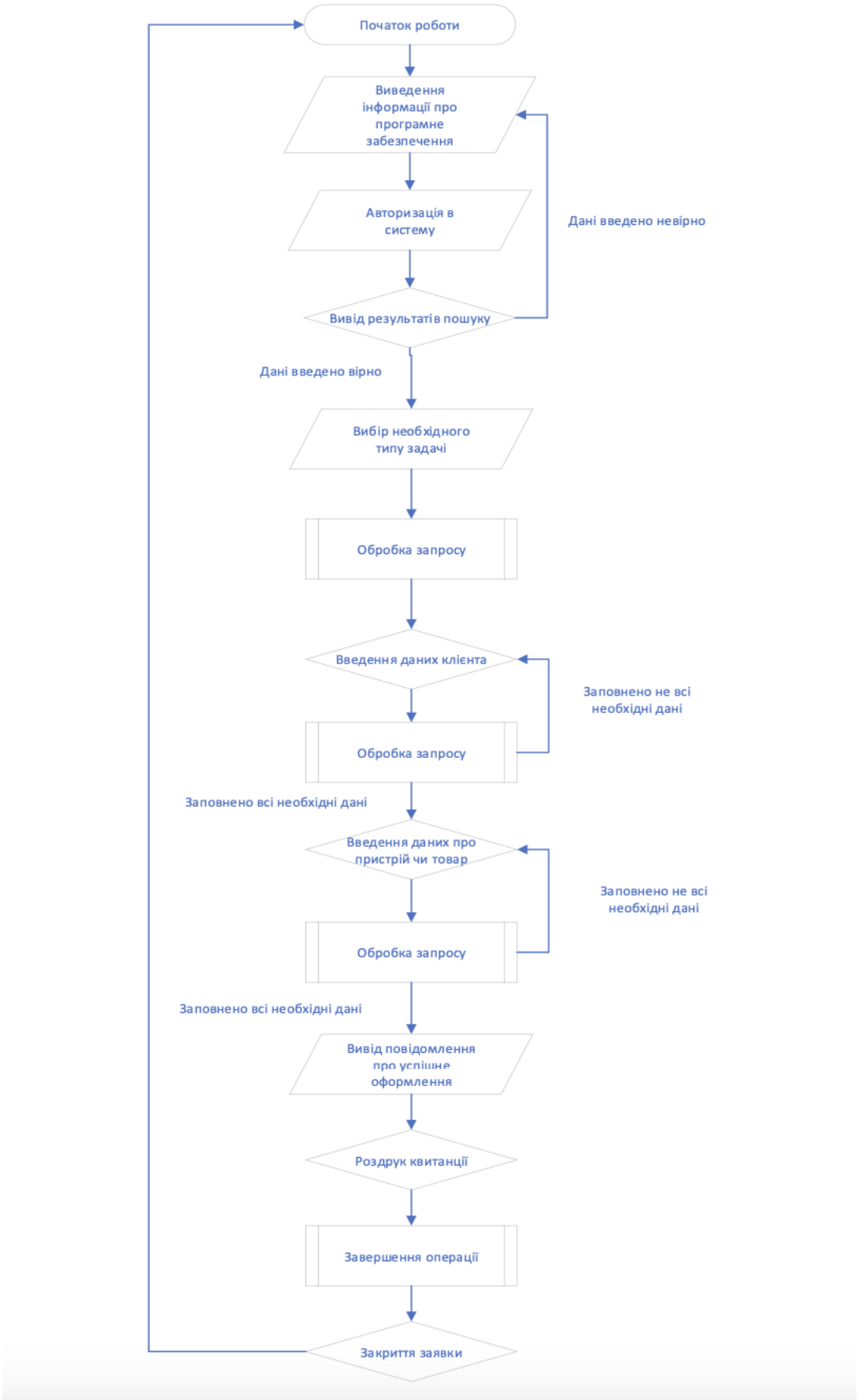

Рис. 4.1. Блок-схема роботи програмного забезпечення

4.3 ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Створене програмне забезпечення для магазину сільгосп техніки має простий та одночасно змістовний інтерфейс. Наприклад, в створеному програмному забезпеченні існує декілька типів основного меню, в якому необхідно обрати об’єкт взаємодії: клієнти, виробник, тип пристроїв, деталі комплекту, мітки, типові несправності, роботи, статуси (Рис. 4.2).

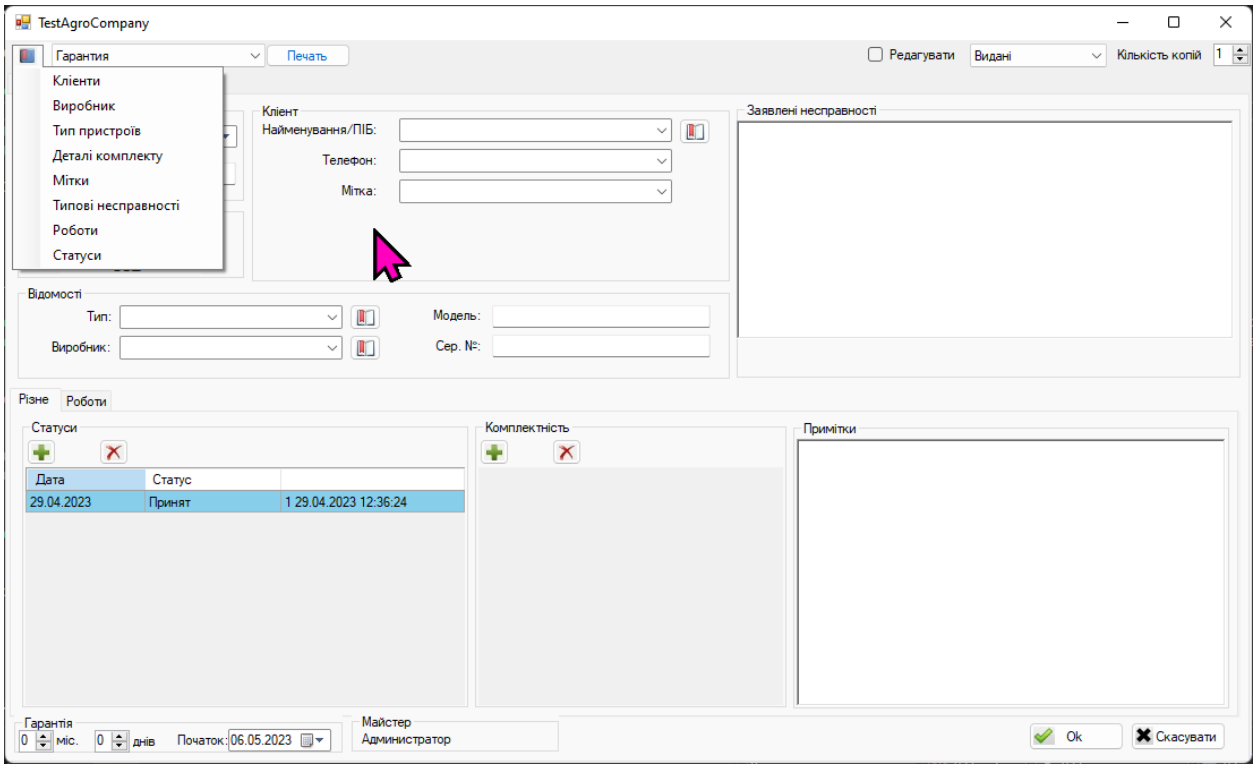

Рис. 4.2. Візуалізація створеного програмного забезпечення

Для зручного користування програмним забезпеченням та прискоренням взаємодії працівників компанії з ним було створено Довідник, в якому можна додати основні або найчастіше використовувані пункти. На Рис. 4.3 представлено візуальне оформлення такого Довідника для додатку найменувань деталей.

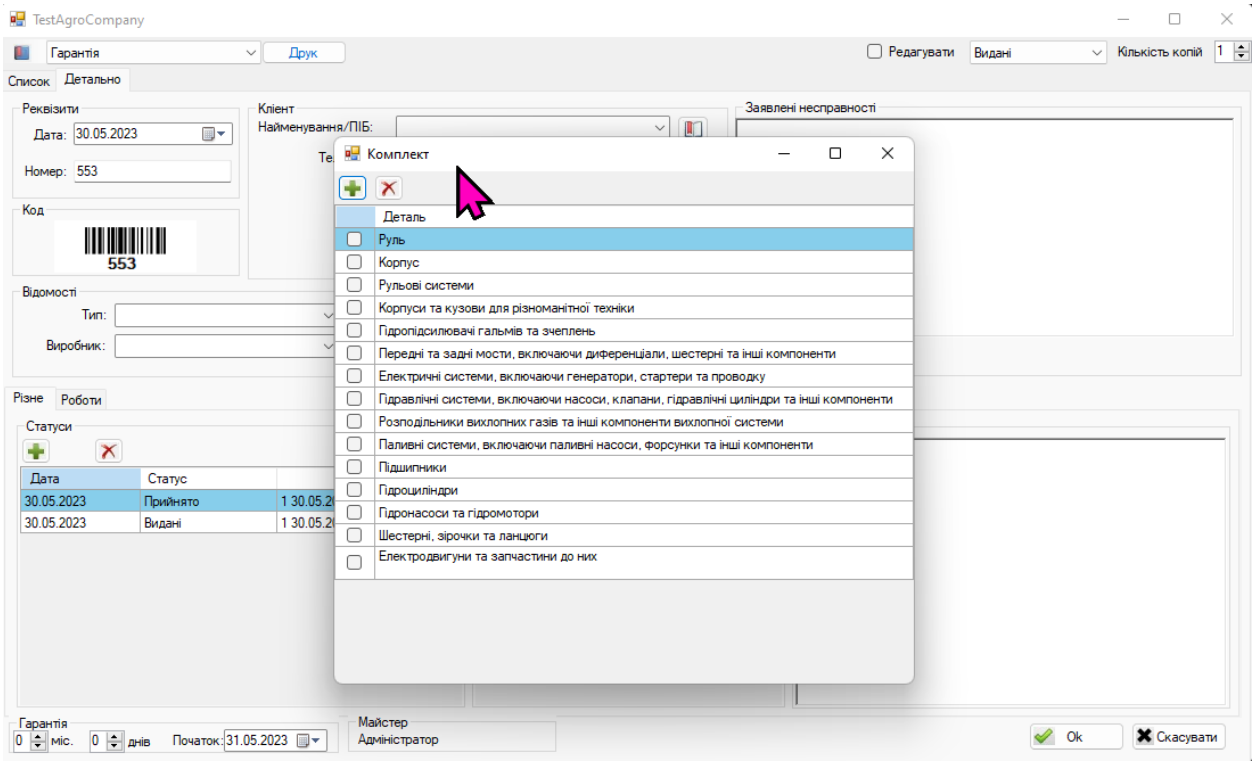

Рис. 4.3. Візуалізація створеного програмного забезпечення

Також в Довідник можна додати найменування брендів товарів для спрощеного пошуку необхідного з них (Рис. 4.4).

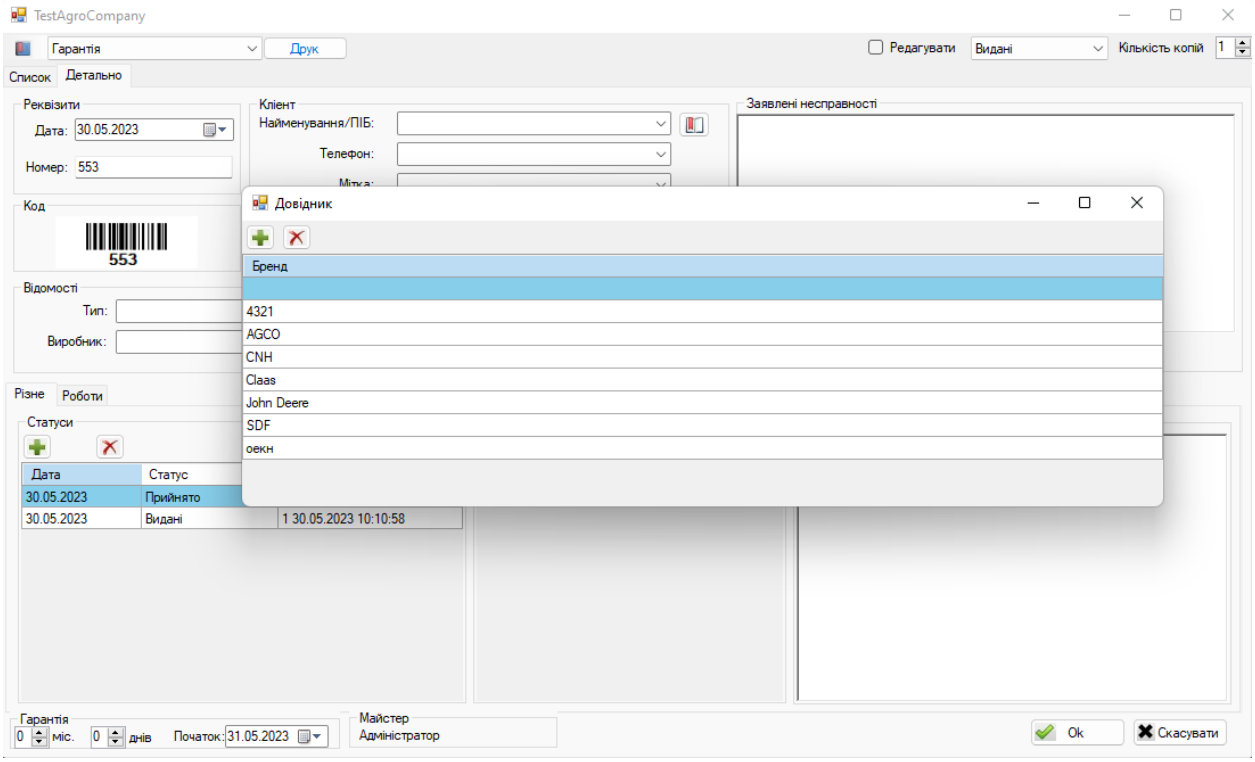

Рис. 4.4. Візуалізація створеного програмного забезпечення

В Довідник також можна додати інформацію про покупців та продавців, партнерів або виробників (Рис. 4.5).

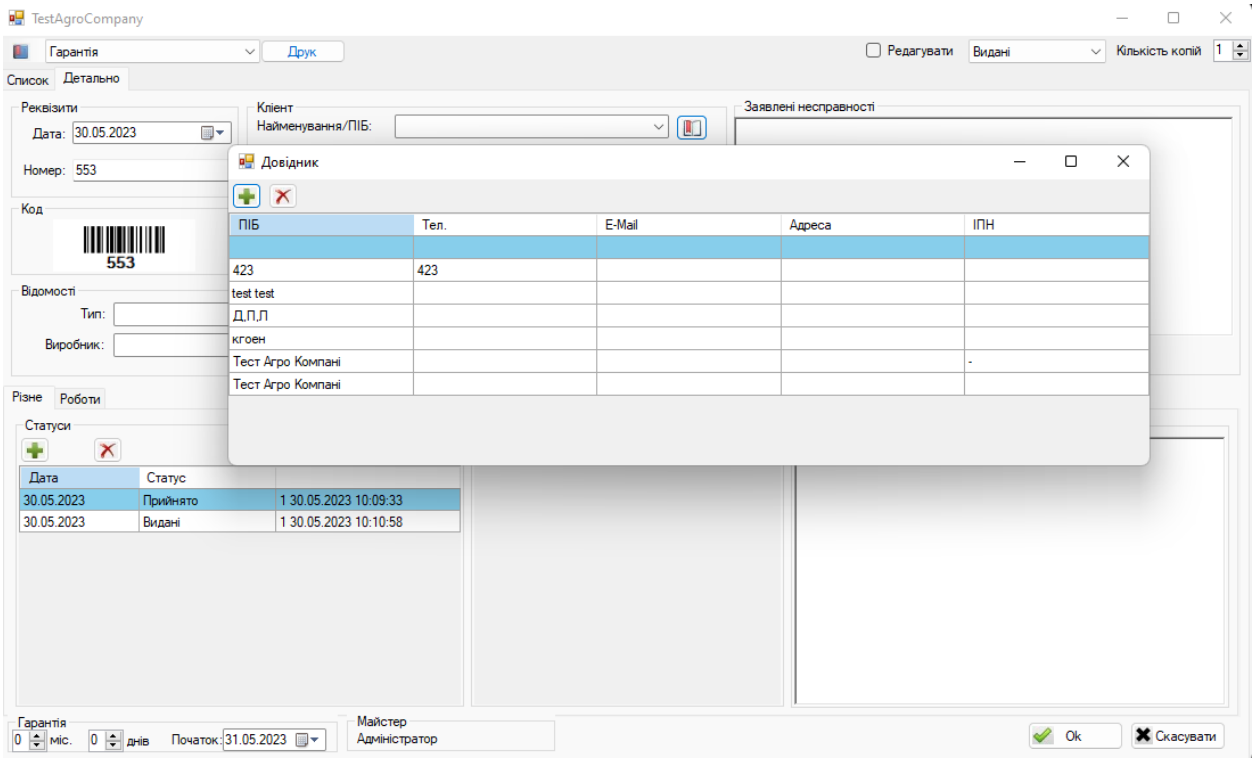

Рис. 4.5. Візуалізація створеного програмного забезпечення

В Довіднику можна коригувати статус для кожної заявки для спрощення процесу взаємодії з покупцями чи потенційними покупцями (Рис. 4.6).

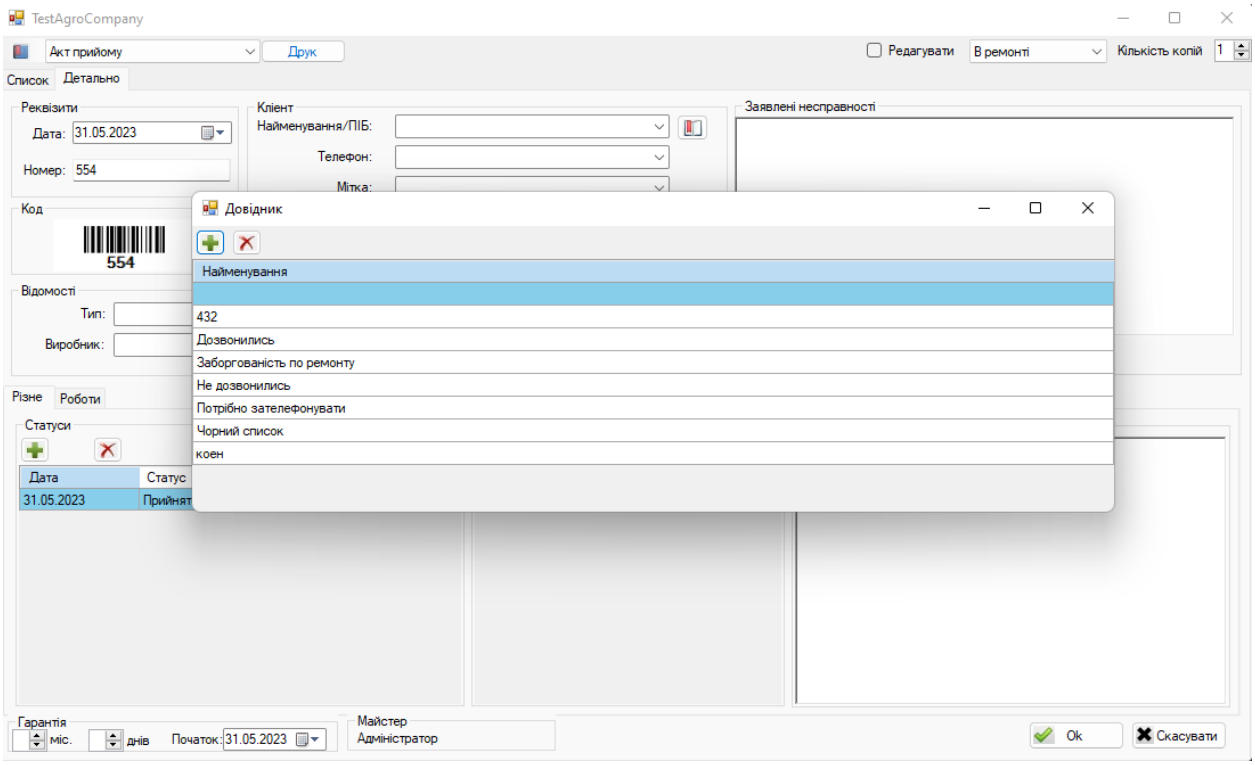

Рис. 4.6. Візуалізація створеного програмного забезпечення

В створеному програмному забезпеченні також можна коригувати статус роботи, що допоможе працівникам сільгосп магазину швидко орієнтуватися в тому, на якому етапі триває обробка того чи іншого запросу клієнта, а також на якому етапі знаходяться прийняті роботи (Рис. 4.7).

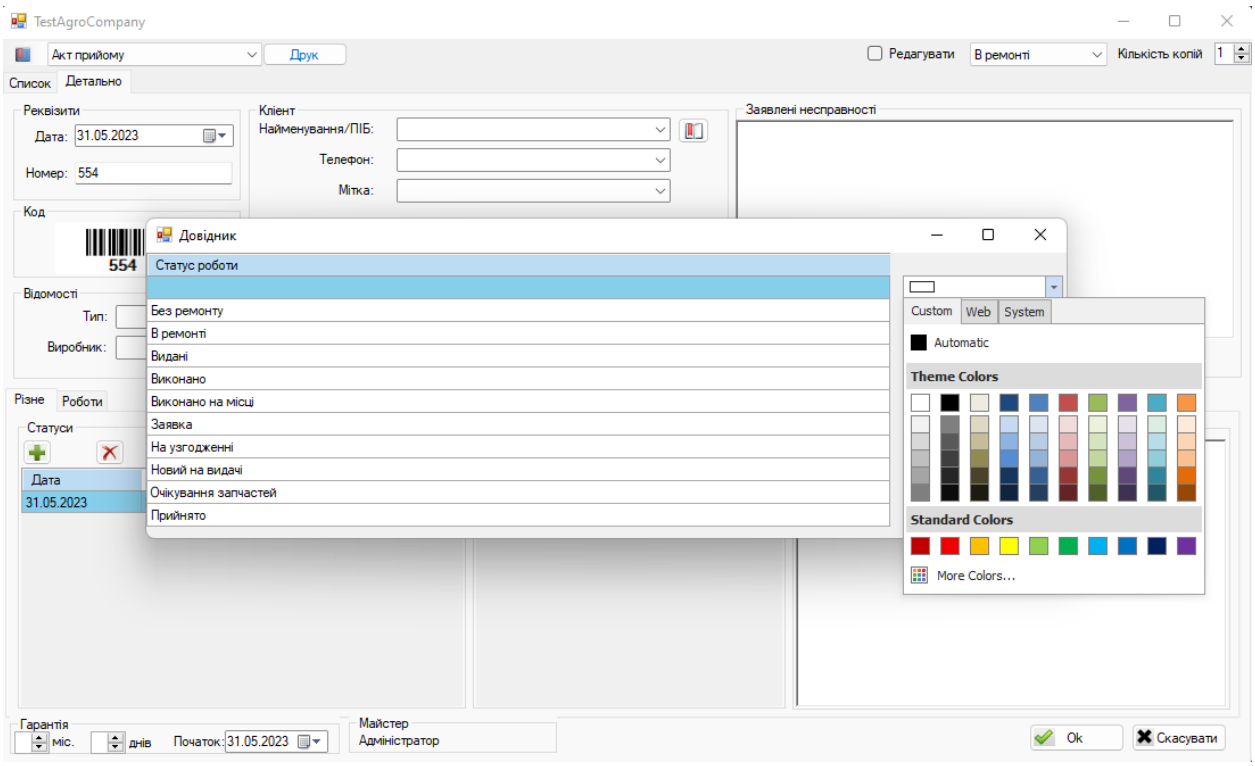

Рис. 4.7. Візуалізація створеного програмного забезпечення

Кожен із статусів можна виділити окремим кольором для швидшої роботи з кожним із клієнтів.

Механізм роботи створеного програмного забезпечення наступний: після отримання заявки від клієнта по телефону чи через спеціальну форму на офіційному сайті магазину працівник створює нову заявку в програмному забезпеченні. Після її обробки необхідно заповнити необхідні дані для випису гарантійного талону на виконані роботи чи проданій товар (Рис. 4.8).

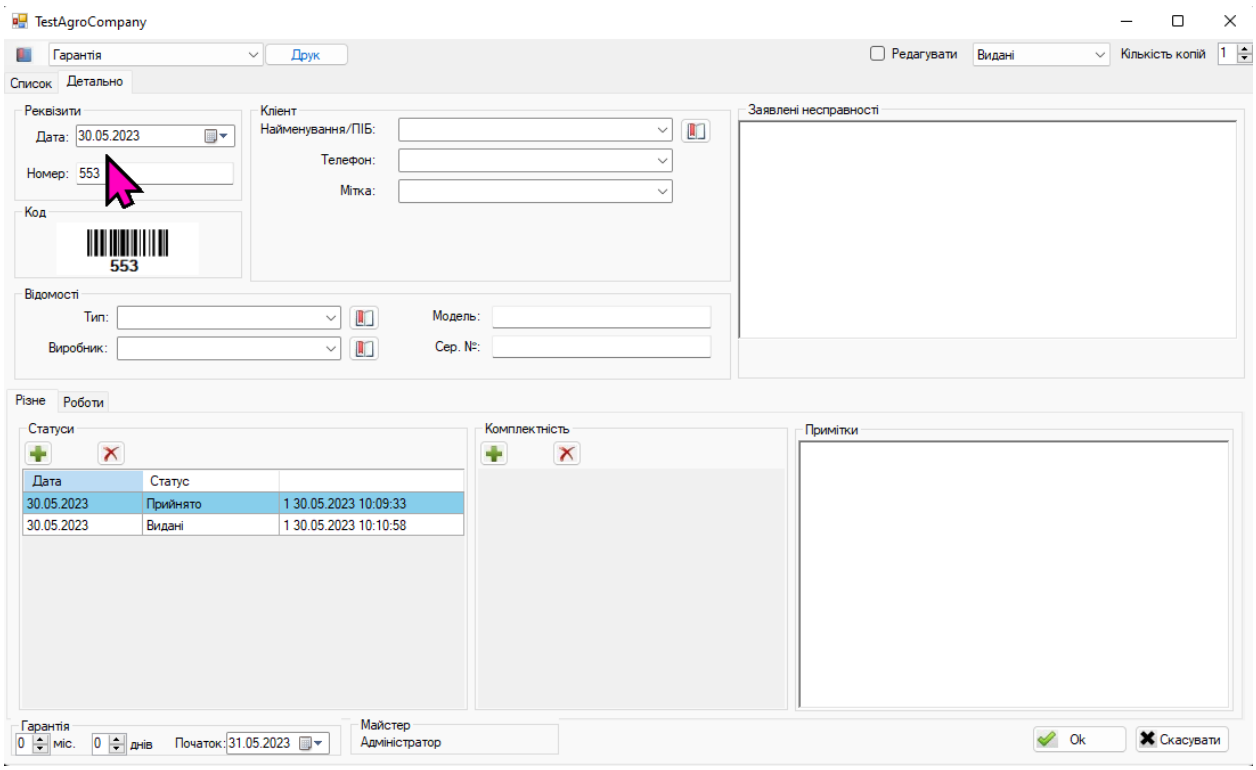

Рис. 4.8. Візуалізація створеного програмного забезпечення

В спеціальному меню програмного забезпечення необхідно обрати назву виробника, тип пристрою, вказати модель та серію (Рис. 4.9).

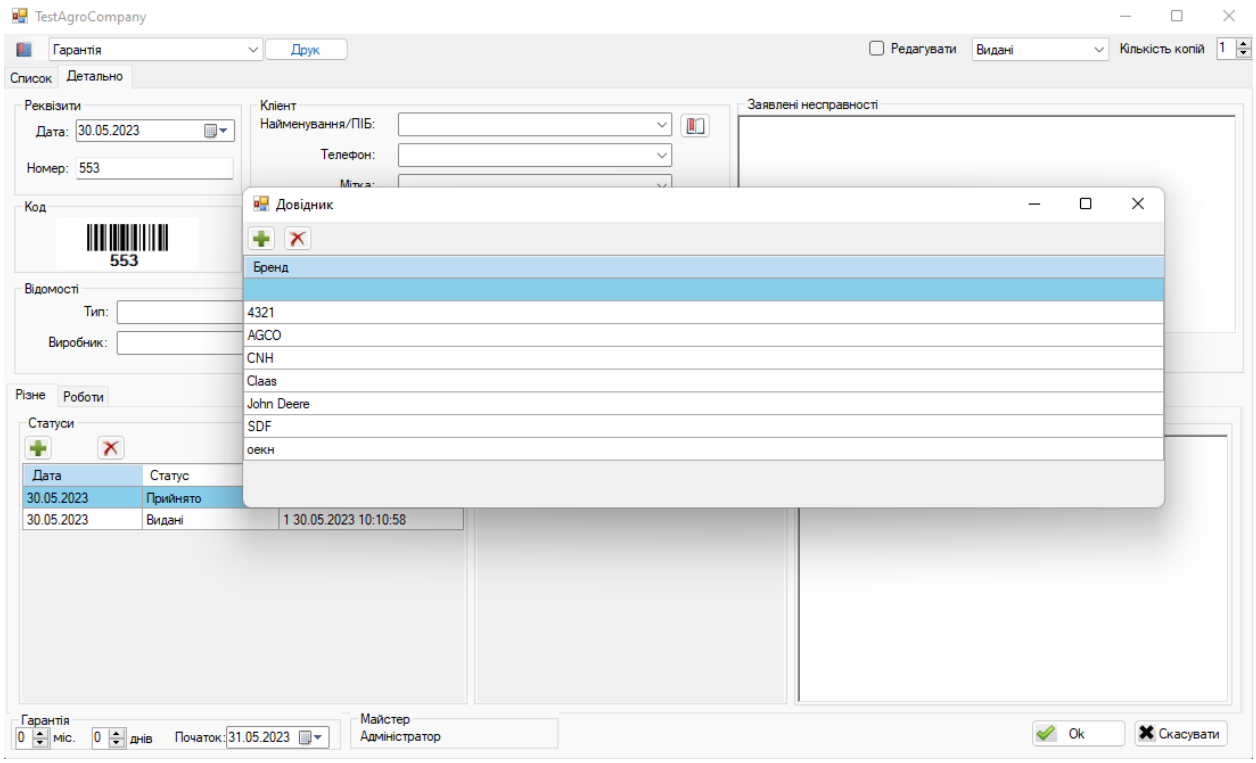

Рис. 4.9. Візуалізація створеного програмного забезпечення

Також необхідно вказати дату початку дії гарантії та строк її дії (Рис. 4.10).

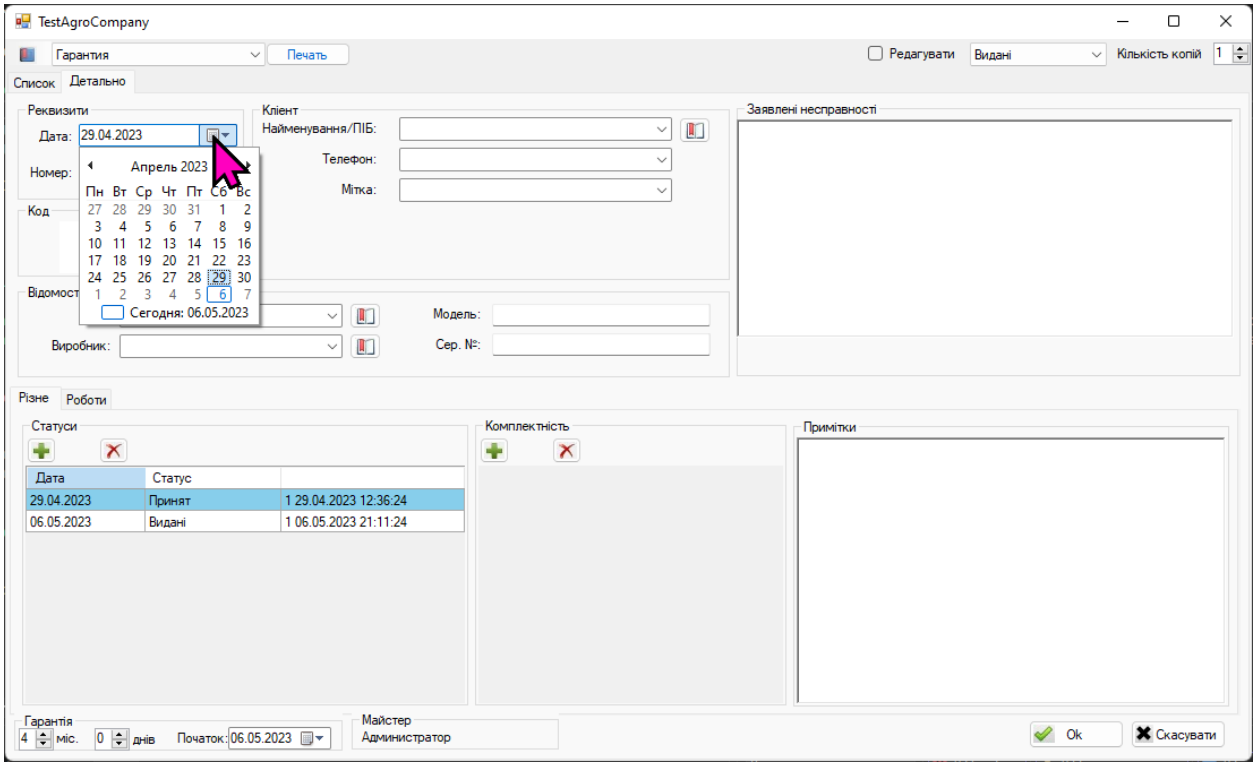

Рис. 4.10. Візуалізація створеного програмного забезпечення

Після заповнення усіх даних програмне забезпечення автоматично створює гарантійний талон, який досить просто роздрукувати та підписати.

Кожна з поступаючих заявок зберігається в спеціальному меню «Акт прийому». Тут вказується перелік усіх заявок, їх номер та тип, назва виробника, контактні дані клієнта, а також статус заявки і особливі примітки до неї (Рис. 4.11).

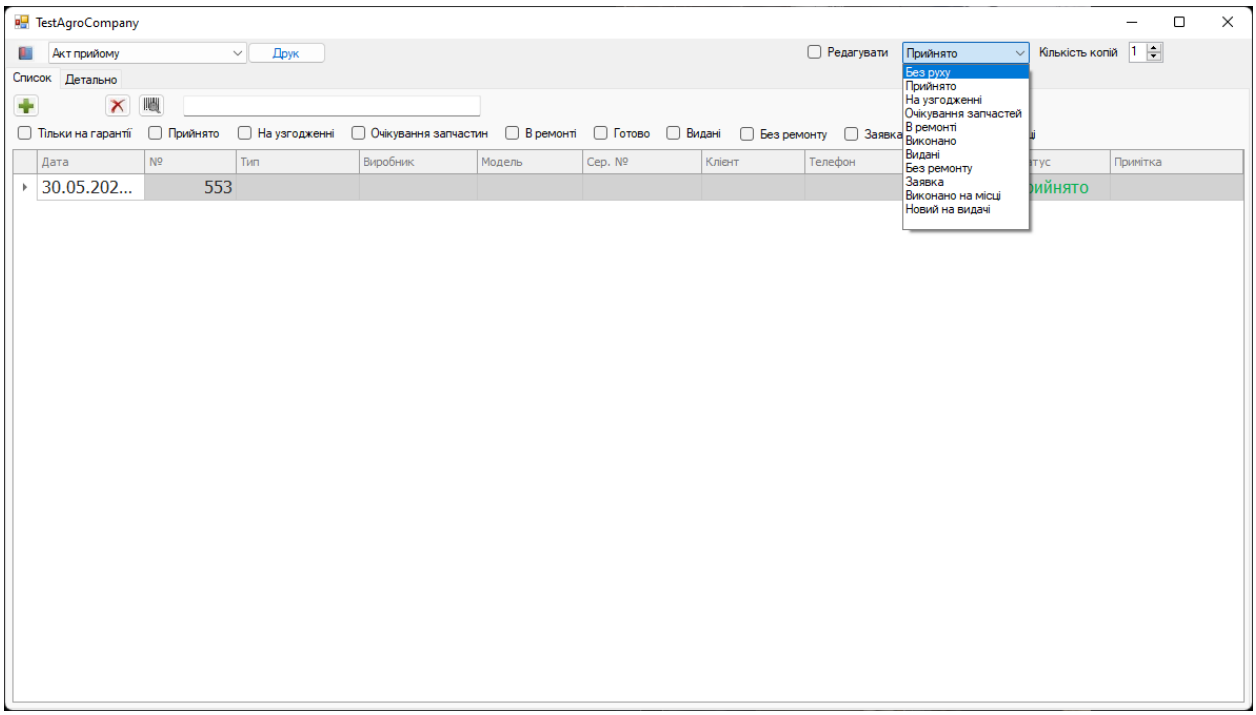

Рис. 4.11. Візуалізація створеного програмного забезпечення

Процес обробки вказаних даних у заявці відбувається в окремому спеціальному запрограмованому додатку (Рис. 4.12).

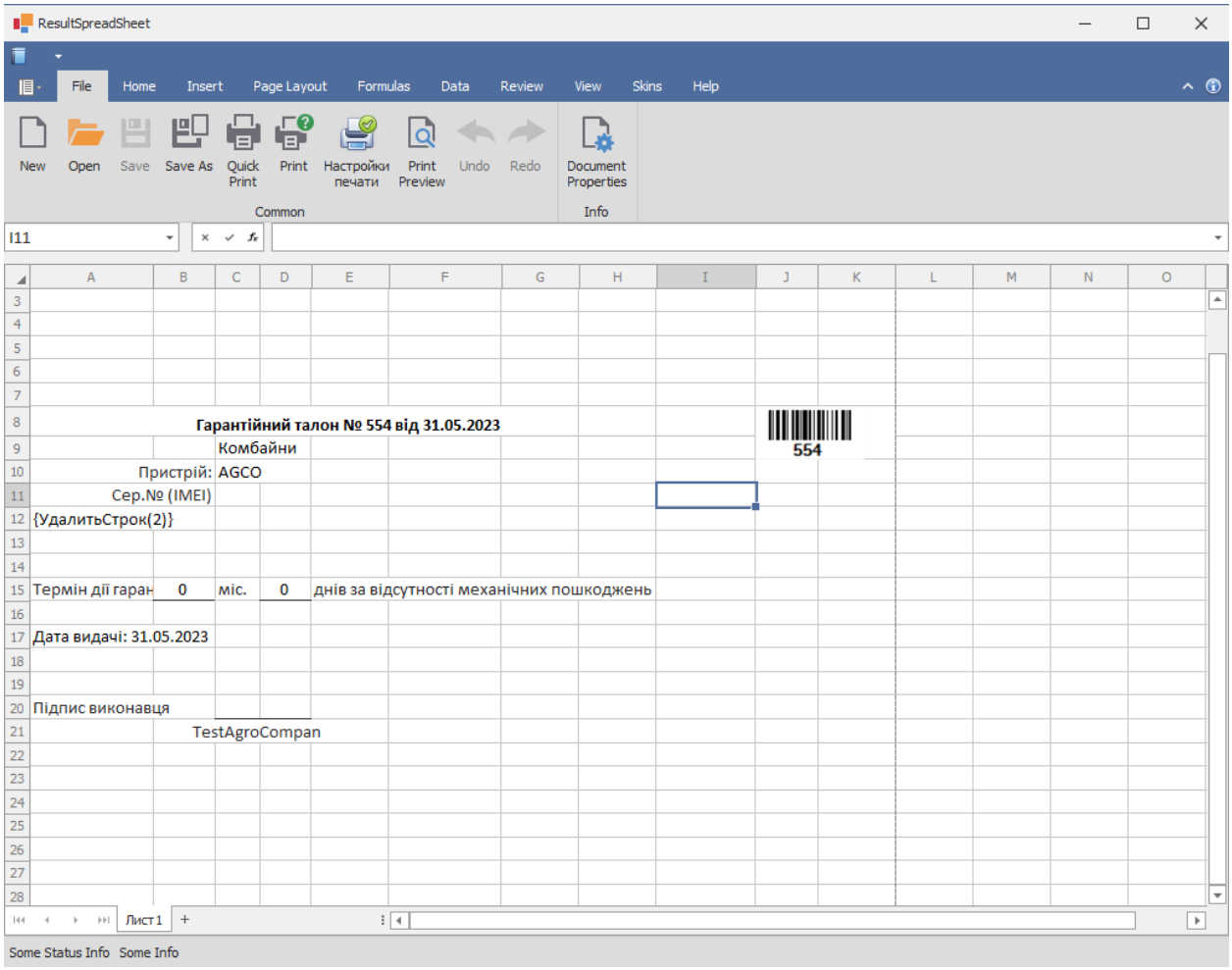

Рис. 4.12. Візуалізація створеного програмного забезпечення

ВИСНОВКИ

Програмне забезпечення для магазинів сільгосп техніки є необхідним для керування операціями, пов'язаними з продажем і обслуговуванням сільгосптехніки. Воно дозволяє автоматизувати багато рутинної роботи, такої як складський облік, управління запасами, ведення фінансової звітності та інвентаризація.

Програмне забезпечення робить управління бізнесом більш ефективним та зменшує кількість помилок, що можуть виникнути в процесі ручної роботи. Воно також дозволяє зберігати дані клієнтів, стежити за їх покупками та поведінкою, що допомагає складати прогнози продажів та виробництва.

Крім того, програмне забезпечення допомагає відслідковувати ремонти та обслуговування техніки, що забезпечує її безперебійну роботу та збільшує її термін служби.

У загальному, програмне забезпечення для магазинів сільгосп техніки дозволяє збільшити ефективність бізнесу, знизити витрати та покращити обслуговування клієнтів.

Розроблене програмне забезпечення допоже у розвитку магазинів сільгосп техніки. З його допомогою працівники зможуть без проблем відслідковувати етапи виконання роботи по тому чи іншому запросу клієнтів, швидко створювати нові заявки, створювати накладний лист чи гарантійний талон в програмному забезпеченні з автоматичним заповненням даних на основі вказаної інформації у створеній заявці та виконувати інші функції, необхідні для успішного функціонування бізнесу.

В ході написання дипломної роботи виконані наступні задачі:

1. описано постановку задачі;
2. описаноінформаційний огляд;
3. переглянуто теоретичний матеріал з теми та навести основні його поняття для використання в програмному забезпеченні;
4. розроблено алгоритм програмного забезпечення;

5. складено блок-схему роботи алгоритму;

6. описано мову програмування та інші технології, що використовувалися при розробці;

7. описано процес реалізації основних етапів створення програмного забезпечення;

8. розроблено програмне забезпечення для магазинів сільсгосп техніки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Програмне забезпечення для автоматизації торгівлі [Електронний ресурс] /. — Електрон. журн. — Техно-трейд, Режим доступу: https://techno-trade.com.ua/ua/g21022745-programmnoe-obespechenie-dlya

2. Автоматизація бізнесу: визначення, приклади, програмне забезпечення та ідеї [Електронний ресурс] /. — Електрон. журн. — Доходність бізнсеу, Режим доступу: https://businessyield.com/uk/business-strategies/business-automation/?currency=USD

3. Види програмного забезпечення [Електронний ресурс] /. — Електрон. журн. — Кафедра АПЕПС ТЕФ КПІ ім. І. Сікорського, Режим доступу: http://apeps.kpi.ua/vidi-programnoho-zabezpechenia

4. Автоматизація процесів обліку і контролю у торгових мережах [Електронний ресурс] /. — Електрон. журн. — ВОСТОК, Режим доступу: https://www.vostok.dp.ua/ukr/infa1/Avtomatizatsiya/avtomatizaciya/

5. Огляд мови C# [Електронний ресурс] /. — Електрон. журн. — ci-sharp.ru, Режим доступу: https://ci-sharp.ru/ukr/Teaching/mova-programmirovaniya-s-obzor-yazika-s.html

6. Переваги і недоліки мови програмування C # [Електронний ресурс] /. — Електрон. журн. — Житомирський національний агроекологічний університет, Режим доступу: http://www.rusnauka.com/21_NPN_2017/Informatica/2_226669.doc.htm

7. C#: що це за мова та де її використовують [Електронний ресурс] /. — Електрон. журн. — robot_dreams, Режим доступу: https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut

8. Програми для роздрібної торгівлі: якими вони повинні бути? [Електронний ресурс] /. — Електрон. журн. — ABM CLOUD, Режим доступу: https://abmcloud.com/programi-dlya-rozdribnoyi-torgivli-yakimi-voni-povinni-buti/

9. Документация по C# [Електронний ресурс] /. — Електрон. журн. — Microsoft, Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/csharp

10. Ольховська О. В. Методичні рекомендації до виконання бакалаврської роботи для студентів за освітньою програмою «Комп’ютерні науки» спеціальності 122 Комп’ютерні науки / О. В. Ольховська, О. О. Черненко. – Полтава : ПУЕТ, 2022. – 71 с.

ДОДАТОК А

Код програмування програмного забезпечення

```
using System;
using System.Windows.Forms;
namespace MKP.PrintDevForms
{
  static class Program
  {
    /// <summary>
    /// Главная точка входа для приложения.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new MainForm());
    }
  }
}
using System;
using System.Globalization;
namespace WindowsFormsApplication2
{
 public static class Addons
 {
  private const string dot = ".";
  private const string comma = ",";
  public static double GetDouble(this string str)
  {
   if (string.IsNullOrWhiteSpace(str))
    return 0.0;
   double result = 0.0;
   str = str.Replace(".", CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator).Replace(",",
CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator);
   if (str.Contains(CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator) && !double.TryParse(str,
out result))
    throw new FormatException("Ошибка во время преобразования в число с плавающей запятой, проверьте
знак разделителя в системе и перезагрузите программу");
   return 0.0;
  }
 }
}
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
 public static class Allert
 {
  public static bool Question(string question) => MessageBox.Show(question, "Підтвердження",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK;
  public static DataTable ToDataTable<T>(this IList<T> data)
  {
   PropertyDescriptorCollection properties = TypeDescriptor.GetProperties(typeof (T));
   DataTable dataTable = new DataTable();
   for (int index = 0; index < properties.Count; ++index)
   {
    PropertyDescriptor propertyDescriptor = properties[index];
    dataTable.Columns.Add(propertyDescriptor.Name, propertyDescriptor.PropertyType);
```

```
   }
   object[] objArray = new object[properties.Count];
   foreach (T component in (IEnumerable<T>) data)
   {
    for (int index = 0; index < objArray.Length; ++index)
     objArray[index] = properties[index].GetValue((object) component);
    dataTable.Rows.Add(objArray);
   }
   return dataTable;
  }
 }
}
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly: AssemblyTitle("WindowsFormsApplication2")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("WindowsFormsApplication2")]
[assembly: AssemblyCopyright("Copyright © 2019")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("74b72b20-577b-4859-a3f3-830cd58bfe02")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: AssemblyVersion("1.0.0.0")]
using dblib;
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
 Подключение баз данных
public static class Connection
 {
  public static string conBD = string.Empty;
  public static string conBDD = string.Empty;
  public static void Init()
  {
   try
   {
#if DEBUG
       if (Environment.MachineName.ToUpper() == "PARHOM")
       {
         Connection.conBD =
@"User=SYSDBA;Password=masterkey;Database=C:\Users\Danil\Desktop\1\BDGame;DataSource=localhost;Port
=3050;Dialect=3;Charset=UTF8;Role=;Connectionlifetime=15;Pooling=true;MinPoolSize=0;MaxPoolSize=50;Pac
ketSize=8192;ServerType=0";
         Connection.conBDD =
@"User=SYSDBA;Password=masterkey;Database=C:\Users\Danil\Desktop\1\BDD;DataSource=localhost;Port=30
50;Dialect=3;Charset=UTF8;Role=;Connectionlifetime=15;Pooling=true;MinPoolSize=0;MaxPoolSize=50;PacketS
ize=8192;ServerType=0";
       }
       else
         throw new Exception("Connection string is not initialized");
#else
    Connection.conBD = Encryption.DecryptToString("bin\\dbset.dll", "5635655899633252");
    Connection.conBDD = Encryption.DecryptToString("bin\\dbbset.dll", "5635655899633252");
#endif
     }
     catch (Exception ex)
   {
    int num = (int) MessageBox.Show("Ошибка при подключении к БД: " + ex.ToString());
    Environment.Exit(0);
```

```
   }
  }
 }
}
namespace WindowsFormsApplication2
{
 public static class Database
 {
  public static bool brandChanged = false;
 }
}
using System.Runtime.Serialization;
namespace WindowsFormsApplication2
{
 [DataContract]
 public class Guarantie
 {
  [DataMember]
  public int monthes;
  [DataMember]
  public int days;
 }
}
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
 public class MyNode : TreeNode
 {
  public int id;
  public MyNode(int id, string name)
  {
   this.Text = name;
   this.id = id;
  }
  public override string ToString() => this.Text;
 }
}
using System.Collections.Generic;
using System.Drawing;
using System.Runtime.Serialization;
namespace WindowsFormsApplication2
{
 [DataContract]
 public class MySettings
 {
  [DataMember]
  public Dictionary<int, Color> devStat;
  public static MySettings settings;
 }
}
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
 internal static class Program
 {
  [STAThread]
  private static void Main()
  {
   Application.EnableVisualStyles();
   Application.SetCompatibleTextRenderingDefault(false);
   Application.Run((Form) new Form1());
  }
```

```
 }
}
namespace WindowsFormsApplication2
{
 public delegate void ReloadSprav(bool ignoreStatus = false);
}
using System;
using System.Collections.Generic;
namespace WindowsFormsApplication2
{
 public class Simple : IComparer<Simple>, IComparable<Simple>
 {
  public object Data;
  public string sData;
  public int iData;
  public string Name { get; set; }

  public int Id { get; set; }
  public Simple Value => this;
  public object[] buffer { get; set; }
  public Simple(string Name, int Id)
  {
   this.Name = Name;
   this.Id = Id;
  }
  public Simple(int Id, string Name)
  {
   this.Name = Name;
   this.Id = Id;
  }
  public override string ToString() => this.Name;
  public string getId() => this.Id.ToString();
  public int GetIntId() => this.Id;
  public void Rename(string newName) => this.Name = newName;
  public override bool Equals(object obj) => obj is Simple simple && simple.Id == this.Id;
  public int Compare(Simple x, Simple y)
  {
   if (x.Id < y.Id)
    return -1;
   return x.Id > y.Id ? 1 : 0;
  }
  public int CompareTo(Simple other) => other == null ? 1 : this.Id.CompareTo(other.Id);
 }
}
namespace WindowsFormsApplication2
{
 public interface Sprav
 {
  void init();
 }
}
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
 public static class TreeWork
 {
  private const string omg = "            ";
  public static void AddTreeItem(
   TreeView t,
   MyNode item,
   string desc,
   int dimension = 0,
```

```
  bool isMainNode = false)
 {
  if (dimension > 0)
   TreeWork.NodeItemAdd(t.Nodes[t.Nodes.Count – 1], item, desc, --dimension, isMainNode);
  else if (!isMainNode)
  {
   t.Nodes.Add((TreeNode) item);
   t.Nodes[t.Nodes.Count – 1].ToolTipText = desc;
  }
  else
  {
   TreeNode node = (TreeNode) item;
   node.NodeFont = new Font(t.Font, FontStyle.Bold);
   t.Nodes.Add(node);
  }
 }
 public static void NodeItemAdd(
  TreeNode t,
  MyNode item,
  string desc,
  int dimension = 0,
  bool isMainNode = false)
 {
  if (dimension > 0)
   TreeWork.NodeItemAdd(t.Nodes[t.Nodes.Count – 1], item, desc, --dimension, isMainNode);
  else if (!isMainNode)
  {
   t.Nodes.Add((TreeNode) item);
   t.Nodes[t.Nodes.Count – 1].ToolTipText = desc;
  }
  else
   t.Nodes.Add((TreeNode) item);
 }
}
}
namespace WindowsFormsApplication2
{
 public delegate void WrkDelegate(MyNode node);
}
```