Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Янишівського В'ячеслава В'ячеславовича*
(ПІБ)

академічної групи *121-19ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення ведення обліку продажу рослин на мові C#*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Удовик І.М.*			
розділів:				
спеціальний	*доц. Удовик І.М.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19ск-1* *Янишівського В'ячеслава В'ячеславовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення ведення обліку продажу рослин на мові C#*

затверджена наказом ректора НТУ «ДП» від . №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Удовик І.М.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Янишівський В.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 81 с., 28 рис., 3 дод., 5 джерел.

Об'єкт розробки: программний додаток для продажу рослин на мові C#

Мета кваліфікаційної роботи: розробка програмного забезпечення для ведення обліку рослин на мові C#

У вступі виконується аналіз сучасного стану проблеми, уточнюється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної галузі та існуючих рішень, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи по створенню застосунку та розраховується час на його створення.

Практичне значення полягає у розробці прорамного забезпечення обліку рослин, що дозволяє оптимізувати процес продажу рослин.

Актуальність програмного продукту визначається наявністю впливу людського фактору на процес продажу рослин та зручністю.

Список ключових слів: ПРОГРАМА, ПРОГРАМНИЙ ДОДАТОК ОБЛІКУ РОСЛИН, ПРОГНОЗУВАННЯ, ЕОМ, ПРИСТРІЙ.

ABSTRACT

Explanatory note: 81 pp., 28 figs., 3 appendices, 5 sources.

Object of development: software application for the sale of plants in C #

Purpose of the qualification work: development of software for keeping records of plants in C #

The introduction analyzes the current state of the problem, specifies the task, the purpose of the qualification work and the field of its application, substantiates the relevance of the topic.

The first section conducts research on the subject area and existing solutions, determines the relevance of the task and the purpose of development, develops the task statement.

The second section selects the platform for development, performs program design and development, describes the algorithm and structure of the system, determines the input and output data, provides characteristics of the parameters of technical means, describes the program.

The economic section determines the complexity of the developed software product, calculates the cost of work to create an application and calculates the time to create it.

Of practical importance is the development of software for plant accounting, which allows to optimize the process of selling plants.

The relevance of the software product is determined by the presence of human influence on the process of selling plants and convenience.

List of key words: PROGRAM, PLANT ACCOUNTING SOFTWARE APPENDIX, FORECASTING, COMPUTERS, DEVICES.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ПД – програмний додаток;

ПЗ – програмне забезпечення;

ІС – інформаційна система;

ОС – операційна система;

СУБД – система управління базами даних;

SQL – Structured Query Language;

UML – Unified Modeling Language.

ЗМІСТ

ВСТУП

Сучасні реалії диктують нові правила ведення бізнесу, і вони кажуть, що неавтоматизованих бізнес швидко стає збитковим і втрачає конкурентоспроможність.

Квіти мають сезонність. Квіти розкуповуються з блискавичною швидкістю на різні свята, а в звичайні дні попит на цей продукт істотно зменшується. Їх продають як окремо, так і в букетах. Вони прикрашаються декоративними елементами і різними пакувальними паперами. Все це впливає на привабливість товару і його кінцеву вартість. Ці особливості квіткового бізнесу вимагають грамотного планування, маркетингу, аналізу і роботи з персоналом. Все це титанічно важко робити вручну і легко з використанням засобів автоматизації

З огляду на вище викладене і обрана тема кваліфікаційної роботи «Розробка програмного забезпечення ведення обліку рослин на мові C#» є актуальною.

Для виконання даної роботи необхідно розробити алгоритм вирішення поставленої задачі.

Метою розроблення кваліфікайної роботи є створення програмного продукту, який забезпечував би робітників належними можливостями роботи та спростив би продаж квітів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Об’єктом дослідження та аналізу предметної галузі є автоматизація процесу продажу рослин для менеджера з метою полегшення праці. Метою проведення аналізу предметної галузі є виявлення, класифікація, формалізація інформації про усі аспекти предметної галузі, що впливають на характеристики кінцевого програмного продукту.

Були впроваджені наступні роботи:

– сбір даних про об’єкт автоматизації та здійснюваних ним видів діяльності;

– вивчення та оцінка якості функціонування об’єкта та здійснюваних ним видів діяльності, виявлення проблем, рішення яких можливе завдяки засобам автоматизації;

– оцінка технічно-економічної, соціальної, практичної доцільності створення програмного продукту;

– виявлення та аналіз вимог користувача до програмного продукту;

– аналіз ринку та потенційних конкурентоспроможних вже існуючих рішень;

– оцінка ефективності прийнятих проектних рішень для створення програмного продукту.

Перейдемо до огляду вже існуючих рішень та результатів дослідження цих систем.

CloudShop — программа для квіткового магазину.

Всі товари легко пробиваються по штрих-коду (актуально для готових букетів і кімнатних рослин), або вручну через вибір товару на касі (спеціальний каталог з фотографіями і описом товару). Для квіткових магазинів, що пропонують послуги флористів, є зручна функція створення наборів. Не потрібно вираховувати

вартість букета, а потім забивати його в систему самостійно. Необхідно просто сформувати комплект, а все інше програма зробить сама.

Кожен продавець буде працювати під власним аккаунтом і нести персональну відповідальність за операції, здійснені в його зміну. Всі проведені операції будуть автоматично відображатися в програмі. Власник магазину без праці зможе відстежувати скільки товару і ким було продано.

Функціонал CloudShop реалізує автоматизацію складського обліку. Всі товари вносяться в онлайн-базу і автоматично списуються при продажу. Ці дані доступні власнику бізнесу в онлайн режимі на смартфоні, для цього потрібен тільки інтернет. Хмарний облік дозволить своєчасно вживати заходів для продажу квітів і не допустити їх в'янення на полицях магазину. Все зроблено максимально зручним для користувача, тому операції оприбуткування і списання виробляються простіше і швидше, ніж при ручному способі.

З таким сервісом значно спроститься маркетинг квіткових товарів. Можна самостійно вносити акції та знижки, тим самим залучаючи клієнтів. Крім того можна без проблем вести базу даних по своїм клієнтам, щоб, наприклад, робити вітальну розсилку з приємними бонусами в честь дня народження. Все розроблено так, щоб було зручно і просто реалізовувати всі маркетингові ідеї.

Усередині сервісу передбачена функція представлення даних в зручній формі для аналізу продажів. Так керівник прямо в своєму смартфоні, навіть будучи закордоном, отримує всі необхідні дані для тактичних і стратегічних рішень.

Окремо варто згадати про те, що в магазинах, що використовують систему CloudShop, істотно підвищується відповідальність працівників, а у випадках з недобросовісними продавцями (частою проблемою квіткових магазинів) все стає явним при першій же звірці.

Спеціальна програма для квіткового магазину істотно спростить всі операції обліку, торгівлі і маркетингу. Крім того, вона відкриє нові горизонти розвитку квіткового магазину за рахунок істотного зниження витрат і прискорення роботи на різних ланках [1].

Отже, підсумовуючи не дивлячись на те, не дивлячись на те, що розроблений автоматизований облік інформації про рослини не є унікальним у своєму роді та має альтернативні варіанти реалізації на програмному ринку (наприклад, програмаCloudShop), він повністю реалізує весь потрібний набір інструментарію, функцій, які э необхідними для комфортного користування ним та коректної роботи виробничих процесів вищезгаданої галузі; на додаток до цього, програмний продукт має додаткові бонусні функції, які не реалізовані в його опонентах на програмному ринку, наприклад, прогнозування різномантних величин, що дозволяє проводити певний аналіз на основі отриманих результатів.

1.2. Призначення розробки та галузь застосування

Програмний продукт, що виконаний для кваліфікаційної роботи, має назву «Розробка програмного забезпечення ведення обліку продажу рослин на мові C#».

Розробка призначена для максимально раціональної та доцільної автоматизації процесів постановки й виконання основних задач та завдань продажу рослин; зберігання та коректної обробки даних, інформації, що фігурують в предметній області; пришвидшення та полегшення роботи з даними; зменшення та мінімалізації прецедентів людського фактору; поєднання всіх функцій в одному сервісі. Программа допоможе робітникам у продажу квітів, мінімізує можливість людського фактору на похибку.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

- освітня програма 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмноо забезпечення ведення обліку продажу рослин на мові C#».

1.4. Постановка завдання

Необхідно розробити програму для ведення обліку інформації про рослини. Програмна система повинна передбачати декілька ролів користувачів: адміністратора та менеджера.

Для менеджера повинні бути доступні наступні функції:

– ведення обліку рослин;
– ведення обліку робітників;
– ведення обліку списаних рослин;
– видачу товарного чеку;
– обробку та аналіз даних, на основі яких здійснюється прогнозування;
– оформлення замовлення про рослини.

Для адміністратора повинні бути доступні наступні функції:

– збереження історії дій робітників та її перегляд;
– додавання, видалення менеджерів.

Також ситема повинна:

– містити детальний опис інструкції користувача;
– перевіряти усі введені дані на коректність
– видавати повідомлення у випадку помилкового вводу.

Для ведення обліку рослин повинна зберігаться наступна інформація: код товару, найменнування, категорія, група, країна походження, ціна за одиницю, кількість;

– дані про рослини можно переглядати, додавати, коригувати, списувати. Списання даних про рослини можливе тільки по одному запису;

Ведення обліку робітників:

– ведення даних про робітників. Дані про робітників повинні містити наступну інформацію: номер робітника, пароль, ПІБ робітника, посада, дата народження, домашння адреса, контактний телефон, електрона пошта;

– дані про робітників можно переглядати усім, окрім паролю, а додавати, коригувати дані всіх, видаляти тільки адміністратору. Робітник може коригувати свої дані. Видалення даних про робітників можливе тільки по одному запису;

Оформлення замовлення про рослини:

– перегляд користувачем каталогу рослин з організацією пошуку по параметрах: код товару, найменнування, країна походження, ціна за одиницю, кількість з можливою комбінацією параметрів пошуку;

– клієнт обирає потрібну рослину, а менеджер оформлює замовлення;

– дані про замовлення повинні містити наступну інформацію: код товару, код менеджера, дата замовлення, ціна за одиницю, кількість товару, сумма.

2 типи користувачів: адміністратор та менеджер:

При вході у програмний додаток потрібно буде ввести номер робітника та пароль. У кожного з користувачів свій рівень доступу.

Видача товарного чеку при оформленні замовлення при наявності принтера.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт має дотримуватися наступних функціональних вимог:

Программа автоматично повинна зберігати інформацію про виконані дії будь-якого менеджера. Цю інформацію може продивитися потім адміністратор, а менеджер може подивитися свою історію. Інформація про історію повина містити наступну інформацію: номер запису, номер робітника, дату та час виконаної дій, інформація про виконані дії.

Ведення обліку списаних рослини.

Рослини які зів'яли, не користуються попитом, після втрати товарного вигляду, втрата первинних властивостей або інше вони списуються, а також вказати кількість списаних рослин. Інформація про списання повина містити наступну інформацію: код рослини, кількість, дата списання, причина списання.

Обробка та аналіз даних, на основі яких здійснюється прогнозування:

– кількість проданих рослин;

– кількість проданих рослин, певним менеджером.

Прогноз повиний виконуватися на день вперед, при цьому менеджер повиний обрати період вихідних даних.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно реалізувати наступні вимоги:

– валідація введених даних: перевірка на відповідність типів, на введення обов’язкових полів даних, валідація діапазонів;

– відмово стійкість (при виникненні помилок програма не повинна припиняти своєї роботи та повідомляти про помилки клієнтській частині).

1.5.3. Вимоги до складу та параметрів технічних засобів

Відштовхуючись від того, що серверна частина додатку працюватиме з потенційно великими обсягами інформації та локальною базою даних, виконуватиме певні обчислення, вибірку даних, звернення до сторонніх ресурсів, були визначені наступні вимоги до конфігурації ПК:

– центральний процесор Intel Pentium, AMD A4 APU або AMD A6 APU;

– оперативна пам’ять 2 ГБ і більше;

– об’єм пам’яті на жорсткому диску 120 Мб і більше;

– операційні системи Windows 7, Windows 8, Windows 10, Windows 11.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмний продукт потребує від користувача мати середовище з такими складовими:

– операційна система Windows.

Серверна частина програмної платформи матиме мову програмування C#, фреймворку Entity Framework та інших додаткових NuGet-пакетів.

Для розробки програмного продукта необхідна наявність наступних програм та систем:

– Visual Studio 2019;

– SQLite;

– GitHubDesktop.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути програма, сумісна з будь-яким клієнтом та працююча на будь-якій платформі, для автоматизації бібліотечної системи, адаптованою під користувача, інтуїтивно зрозуміла та зручна у використанні.

Експлуатаційне призначення – автоматизація та спрощення ручної праці людини під час введення обліку книг, читачів та формулярів. Розроблена система дозволяє звести використання паперу, значно зменшує витрати часу та людських ресурсів для аналізу отриманих даних, їх упорядкування, збереження, а також сортування, редагування, пошуку та додавання.

Функціональне призначення – надати можливість користувачеві у ролі бібліотекара додавати нових читачів та нові книги, швидко оформлювати видавати читачу потрібні книги, та реєструвати повернення, введення інформації про книги.

2.2. Опис застосованих математичних методів

У програмному продукті було обрано реалізувати прогнозування за допомогою методу найменших квадратів.

Метод найменших квадратів (МНК) є одним із методів регресійного аналізу, який використовується для статистичного оцінювання параметрів регресійної моделі за емпіричними даними. Згідно з цим методом параметри моделі повинні відповідати такому рівнянню регресії, що забезпечує найменше значення суми квадратів відхилень емпіричних даних від тих, що обчислені за рівнянням регресії. Так, з двох різних наближень тієї ж самої емпіричної функції, що задана у вигляді таблиці, кращим вважається те, для якого сума квадратів

відхилення має найменше значення. Для ілюстрації суті методу найменших квадратів можна розглянути рис. 2.1.

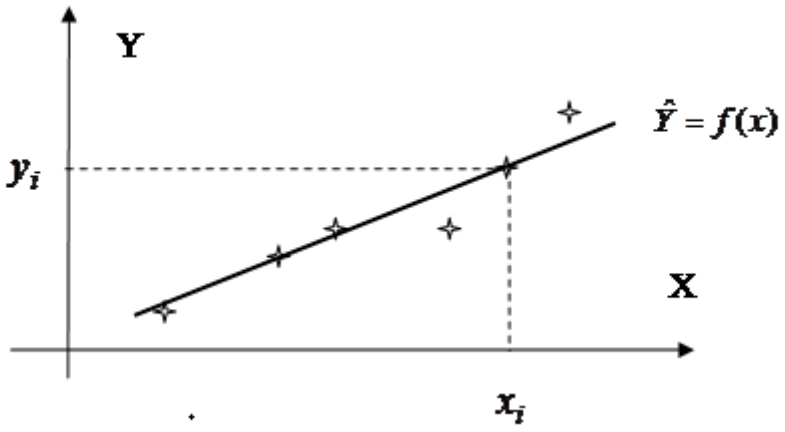


Рис. 2.1. Ілюстрація методу найменших квадратів

Графік функції проходитиме таким чином, щоб різниця між значеннями функції та ординатами емпіричних точок була б якомога менше.

Основи метода найменших квадратів були розроблені Карлом Фрідріхом Гауссом у зв'язку з задачами, що вирішуються теорією помилок, тобто математичною теорією, яка досліджує точність результатів вимірювань. До речі, Гаусс настільки ретельно провів дослідження нормального розподілу випадкових похибок, що їх крива щільності ймовірностей має назву гауссіани.

Узагальнення умов застосування методу найменших квадратів сформульовано у теоремі Гаусса – Маркова.

Моделі, які побудовані з використанням методу найменших квадратів при виконанні умов теореми Гаусса – Маркова, називаються класичними економетричними моделями, а сам метод їх побудови називається 1МНК, тобто однокроковий метод найменших квадратів.

2.3. Опис використаної архітектури та шаблонів проектування

SQLite – це реляційна база даних, сумісна з SQL. На відміну від інших систем на основі SQL, таких як MySQL та PostgreSQL, SQLite не використовує

архітектуру клієнт-сервер. Вся програма міститься в бібліотеці C, яка інтегрована у додатки. База даних стає невід'ємною частиною програми, усуваючи ресурсомісткі автономні процеси.

SQLite зберігає свої дані в одному кроссплатформенном файлі. Оскільки немає виділеного сервера або спеціалізованої файлової системи, "розгорнути" SQLite так само просто, як зв'язати його бібліотеку та створити новий звичайний файл.

Ця простота призвела до масового впровадження SQLite як кращу систему баз даних для додатків і пристроїв, що вбудовуються. Вважається, що загальна кількість розгортань SQLite перевищує всі інші движки баз даних. комбінований тому що він поставляється в комплекті з усіма основними операційними системами, більшістю мов програмування, великим списком вбудованого обладнання та багатьма основними програмними продуктами.

SQLite фокусується на наданні потужної SQL-сумісної бази даних без накладних витрат або залежностей. Як випливає з назви, це легке рішення, яке може працювати практично на всьому, що підтримує C та постійне сховище файлів. Прив'язки доступні найпопулярніших мов програмування високого рівня.

Оскільки бази даних SQLite є простими файлами, їх легко переносити і створювати резервні копії. Відсутність серверного компонента значно спрощує налаштування SQLite, навіть якщо ви не є повноцінним середовищем розробки. Немає тривалого процесу відстеження, перезапуску або перевірки проблем безпеки.

Програми, що використовують SQLite, отримують вигоду від підвищеної відмовостійкості та скорочення часу розробки. Використання бази даних SQLite замість текстових файлів для конфігурації та зберігання уніфікує доступ до даних на всіх пристроях та допомагає підтримувати стабільну продуктивність.

Бази даних містять засоби захисту від пошкодження, яких немає у звичайних файлах. SQLite є атомарним і транзакційним, тому можна уникнути

часткового успіху. Якщо операція транзакції завершується невдало, всі успішні операції також відкочуються, повертаючи базу даних у вихідний стан.

SQLite також стійкий до помилок зберігання та сценаріїв нестачі пам'яті. Бази даних можуть відновлюватися після повного збою системи та відключення електроенергії, що допомагає захистити дані. Кодова база покрита повсюдним набором тестів зі 100% покриттям.

2.4. Опис використаних технологій та мов програмування

Серверна частина розроблюваної платформи написана мовою C#. з використанням фреймворку ASP.NET Core, вона є WebAPI-службою. В якості бази даних використовується MS SQL (Local DB), також використаний Entity Framework для спілкування між серверною програмною частиною та базою даних. Також присутні різноманітні NuGet-пакети. У якості UI був використано WP, технологія для відображення інтерфейсу на OS Windows;

В якості IDE обрана Visual Studio 2019 Community, веб-серверу – IIS. Система управління версій – git, робота з яким виконувалася завдяки додатку GitHubDesktop.

C# — проста, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Важлива особливість таких компонентів — це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. C# надає мовні конструкції, безпосередньо підтримують таку концепцію роботи. Завдяки цьому C# відмінно підходить для створення і застосування програмних компонентів.

Ось лише кілька функцій мови C#, що забезпечують надійність і стійкість додатків: прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень дає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не

дозволяє звертатися до неініціалізованих змінним, виходити за межі масиву або виконувати неконтрольоване приведення типів.

У C# існує єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці [2].

Класи є основним типом в мові C#. Клас являє собою структуру даних, яка об'єднує в собі значення (поля) і дії (методи і інші функції-члени). Клас надає визначення для динамічно створюваних екземплярів класу, які також іменуються об'єктами. Класи підтримують механізми успадкування та поліморфізму, які дозволяють створювати похідні класи, що розширюють і уточнюють визначення базових класів.

Нові класи створюються за допомогою оголошень класів. Оголошення класу починається з заголовка, в якому вказані атрибути і модифікатори класу, ім'я класу, базовий клас (якщо є) і інтерфейси, реалізовані цим класом.

Об’єкти класів створюються за допомогою оператора new, який виділяє пам'ять для нового екземпляра, викликає конструктор для ініціалізації цього об’єкта і повертає посилання на екземпляр.

Займана об'єктом пам'ять автоматично звільняється, коли об'єкт стає недоступний. У C# немає ні необхідності, ні можливості звільняти пам'ять об'єктів явно [3].

Нижче перераховані види членів, які можуть міститися в класі.

- константи;
- поля;
- методи;
- властивості;
- індексатори;

– події;

– конструктори;

– методи завершення.

Кожен член класу має певний рівень доступності. Він визначає, з якої області програми можна звертатися до такого Члена СОТ. Існує шість рівнів доступності. Вони коротко описані нижче.

– public (Доступ не обмежений);

– protected (Доступ можливий з цього класу і з класів, успадкованих від нього);

– internal (Доступ обмежений тільки поточної складанням (.exe, .dll);

– protectedinternal (Доступ можливий з цього класу і з класів, успадкованих від нього, або класами в тій же збірці);

– private (Доступ можливий тільки з цього класу);

– privateprotected (Доступ обмежений містить класом або класами, які є похідними від типу в тій же збірці).

Визначення класу може задати набір параметрів. Список імен параметрів типу вказується в кутових дужках після імені класу. Параметри типу можна використовувати в тілі класу в визначеннях, що описують члени класу.

Клас успадковує члени базового класу. Спадкування означає, що клас неявно містить всі члени свого базового класу, за винятком конструкторів примірника, статичних конструкторів і методів завершення базового класу. Похідний клас може доповнити успадковані елементи новими елементами, але він не може видалити визначення для успадкованого члена.

Поле є змінною, пов'язаної з певним класом чи примірником класу [4].

Поле, оголошене з модифікатором static, є статичним. Статичне поле визначає строго одне місце зберігання. Незалежно від того, скільки буде створено реалізації відповідного класу, існує тільки одна копія статичного поля.

Метод — це член, який реалізує обчислення або дія, яка може виконувати об'єкт або клас. Доступ до статичних методів здійснюється через клас. Доступ до методів екземпляра здійснюється через екземпляр класу.

Для методу можна визначити список параметрів, які представляють передані методу значення або посилання на змінні, а також повертається тип, який задає тип значення, що обчислюється і повертається методом. Якщо метод не повертає значень, для нього встановлюється повертається тип void.

Параметр значення використовується для передачі вхідних аргументів. Параметр значення зіставляється з локальної змінної, яка отримає початкове значення з значення аргументу, переданого в цьому параметрі. Зміни параметра значення не впливають на аргумент, переданий для цього параметра.

Параметри значення можна зробити необов'язковими, вказавши для них значення за замовчуванням. Тоді відповідні аргументи можна не вказувати.

Параметр виведення використовується для передачі аргументів за посиланням. Він схожий на контрольний параметр, однак не вимагає явно привласнювати значення аргументу, наданого викликає об'єктом. Щоб оголосити параметр виводу, використовуйте модифікатор out.

Усередині методу масив параметрів повністю ідентичний звичайному параметру типу масив. Але зате при виклику методу, що використовує масив параметрів, йому можна передати або один аргумент типу масив, або будь-яку кількість аргументів типу елемент для масиву параметрів. В останньому випадку екземпляр масиву автоматично створюється і ініціалізується за заданими аргументами.

C# вимагає, щоб локальної змінної було явно присвоєно значення, перш ніж можна буде отримати це значення.

Метод може використовувати інструкцію return, щоб повернути управління зухвалому об'єкту. Якщо метод повертає void, в ньому не можна використовувати інструкцію return з виразом. У методі, вихідне значення якого має будь-який інший тип, інструкції return повинні містити вираз, яке обчислює значення, що повертається.

Як і щодо полів і методів, C# підтримує властивості екземпляра і статичні властивості. Статичні властивості оголошуються з модифікатором static, а властивості екземпляра - без нього [5].

2.5. Опис структури програми та алгоритмів її функціонування

Спроектована база даних містить 5 пов’язаних між собою таблиць. Нижче можна побачити фізичну модель створюваної бази даних (рис. 2.2).

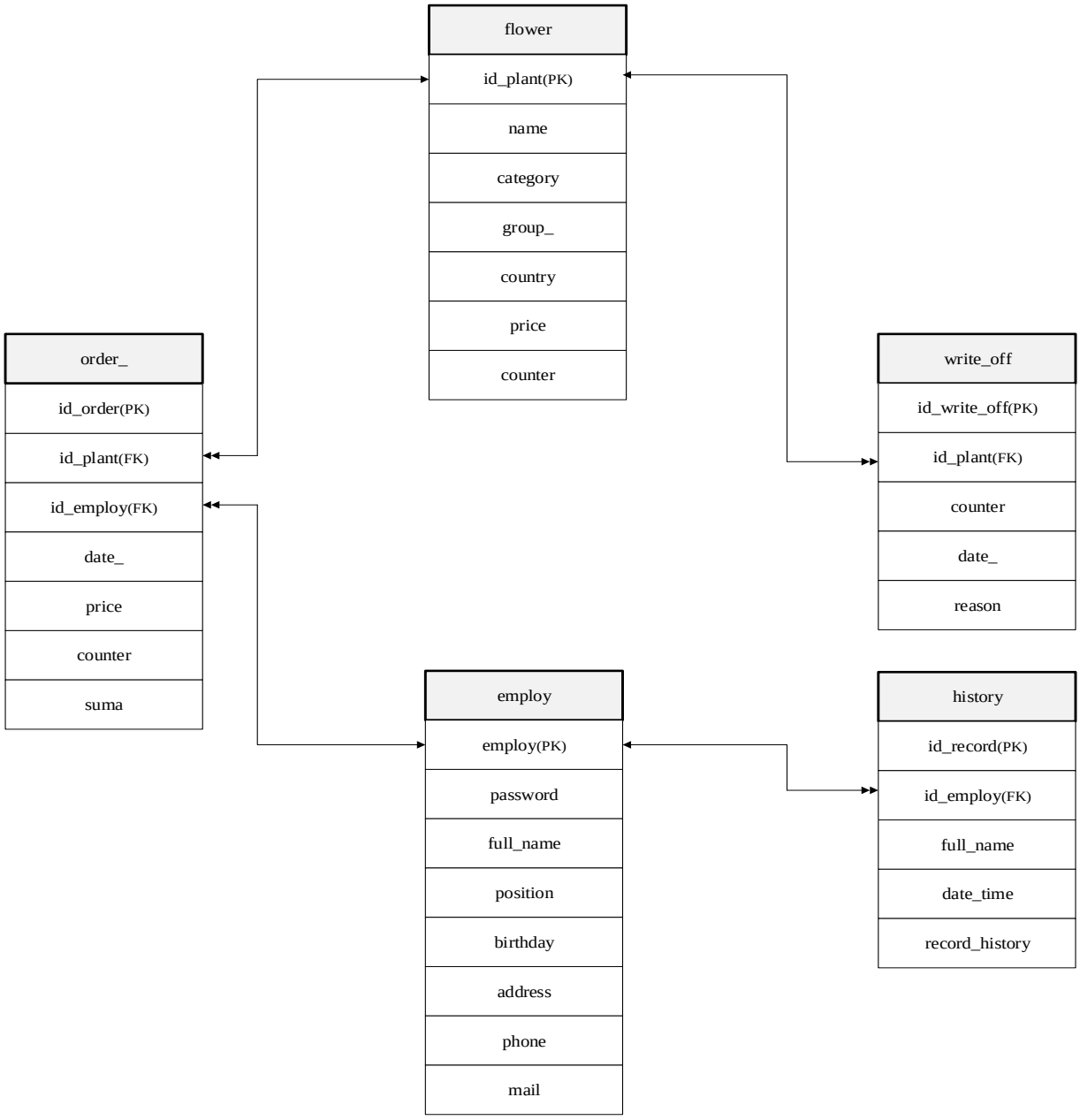


Рис. 2.2 – ER-діаграма БД

Структури таблиць БД кваліфікаційної роботи зображені на рисунках 2.3-2.7.

Имя таблицы: plant ☐ WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id_plant	INTEGER							NULL
2	name	VARCHAR (999)							NULL
3	categor	VARCHAR (999)							NULL
4	group_	VARCHAR (999)							NULL
5	country	VARCHAR (999)							NULL
6	price	VARCHAR (999)							NULL
7	counter	VARCHAR (999)							NULL

Рис. 2.3 – Структура таблиці plant (дані про рослини)

Имя таблицы: employ ☐ WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id_employ	INTEGER							NULL
2	password	VARCHAR (999)							NULL
3	full_name	VARCHAR (999)							NULL
4	position	VARCHAR (999)							NULL
5	birthday	VARCHAR (999)							NULL
6	address	VARCHAR (999)							NULL
7	phone	VARCHAR (999)							NULL
8	mail	VARCHAR (999)							NULL

Рис. 2.4 – Структура таблиці employ (дані про робітників)

Имя таблицы: order_ ☐ WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id_order	INTEGER							NULL
2	id_plant	INTEGER							NULL
3	id_employ	INTEGER							NULL
4	date_	VARCHAR (999)							NULL
5	price	VARCHAR (999)							NULL
6	counter	VARCHAR (999)							NULL
7	suma	VARCHAR (999)							NULL

Рисунок 2.5 – Структура таблиці order_ (дані про замовлення)

Имя таблицы: history WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id_record	INTEGER							NULL
2	id_employ	VARCHAR (999)							NULL
3	date_time	VARCHAR (999)							NULL
4	record_history	VARCHAR (999)							NULL

Рис. 2.6 – Структура таблиці history (дані про історію дій робітників)

Имя таблицы: write_off WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id_write_off	INTEGER							NULL
2	id_plant	VARCHAR (999)							NULL
3	counter	VARCHAR (999)							NULL
4	date_	VARCHAR (999)							NULL
5	reason	VARCHAR (999)							NULL

Рис. 2.7 – Структура таблиці write_off (дані про списані рослини)

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Для запобігання введення некоректних даних було встановлено спеціальні властивості компонентів та створено відповідні перевірочні функції, що забезпечують правильну роботу програмної системи.

На формі ADDUPDATEPLANT розташований компонент, який натискается при заповнені усіх полей.

Якщо вони не заповнені, або не всі заповнені, або некорректно заповнені поля, користувачеві не буде дана можливість натискати на кнопку, зображених на рисунку 2.8

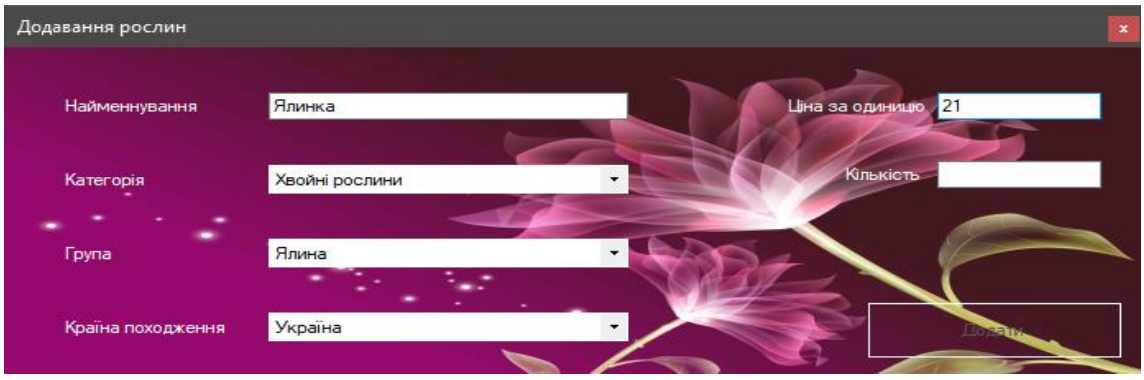


Рис. 2.8 — Відлючена кнопка, бо не всі поля не заповнені

Також на формі відбувається перевірка, яка відповідає за те, щоб контактні дані не повторювалися. На рисунку 2.9 зображено повідомлення при повторюванному вводу.

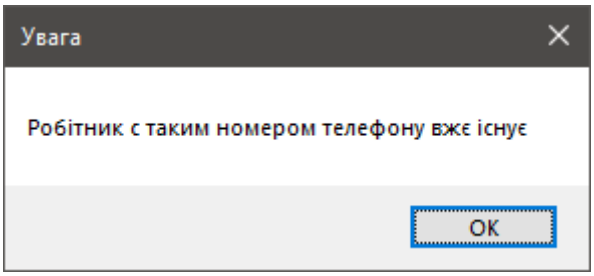


Рис. 2.9 — Повідомлення про повторення номеру телефону

Використав обробник виключних ситуацій для того, щоб при невдалому виповненню кода программа аварійно не завершалась. На рисунку 2.10 зображено повідомлення при попадані в блок виключення.

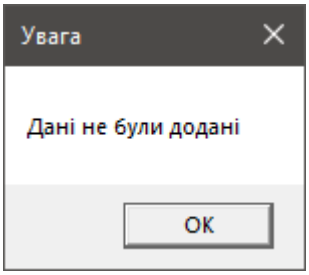


Рис. 2.10 — Повідомлення про помилку виконання запиту

Використав на формі ADDUPDATEPLANT компонент ComboBox для того щоб користувачу не потрібно було вводити дані, а тільки вибрати, тим самим забезпечити безпечний та зручний ввод данних. Використання ComboBox зображено на рисунку 2.11.

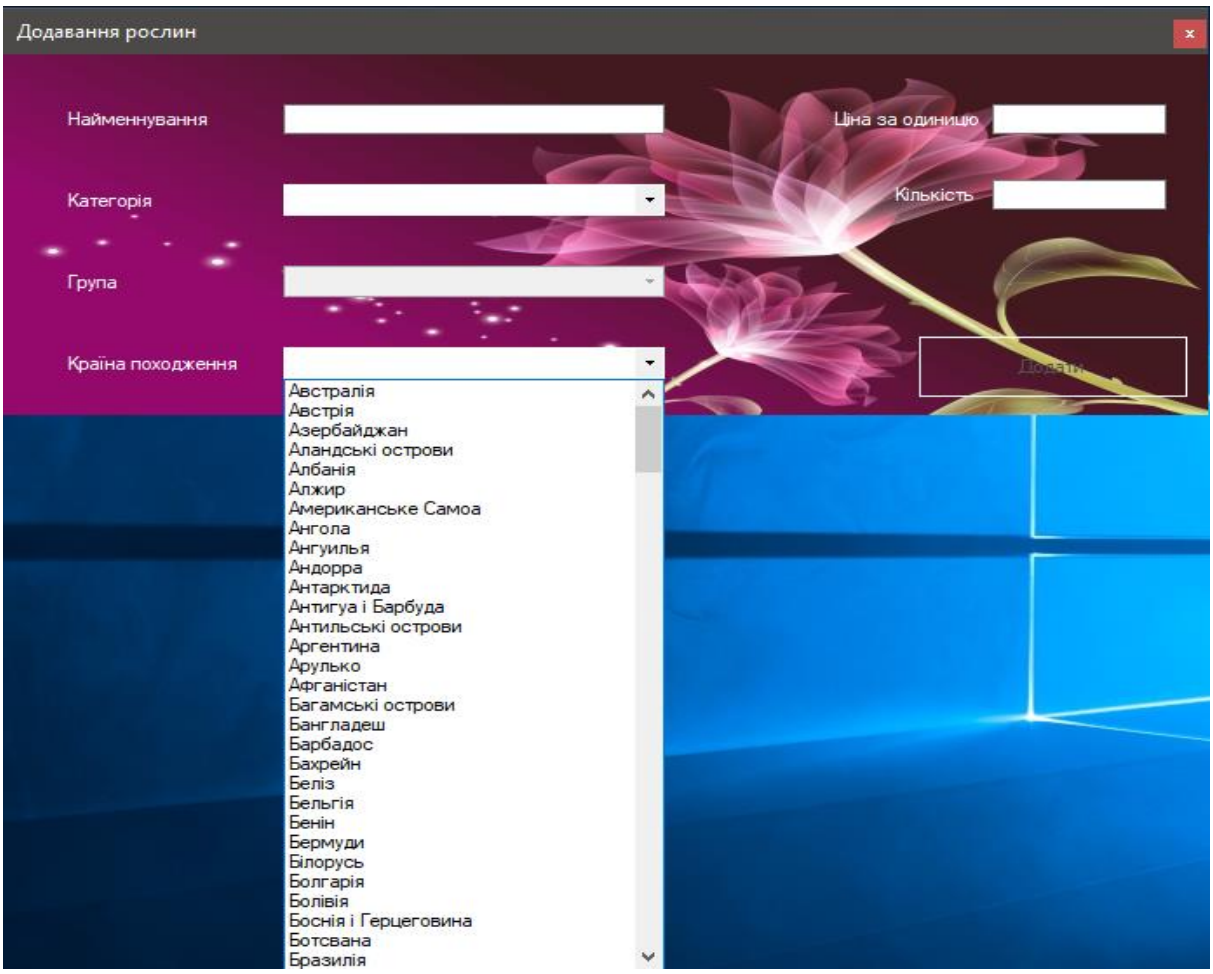


Рис. 2.11 — Використання ComboBox

Використав на формі ADDUPDATEEMPLOY компонент MaskedTextBox для того щоб користувачу легше було вводити номер телефону, тим самим забезпечити безпечний та зручний ввод данних. На рисунку 2.12 зображено використання MaskedTextBox.

Рис. 2.12 — Використання MaskedTextBox

Використав на формі ADDUPDATEEMPLOY компонент DateTimePicker для того щоб користувачу не потрібно було вводити дані, а тільки обрати значення з календаря, тим самим забезпечити безпечний та зручний ввод данних. На рисунку 2.13 зображено використання DateTimePicker.

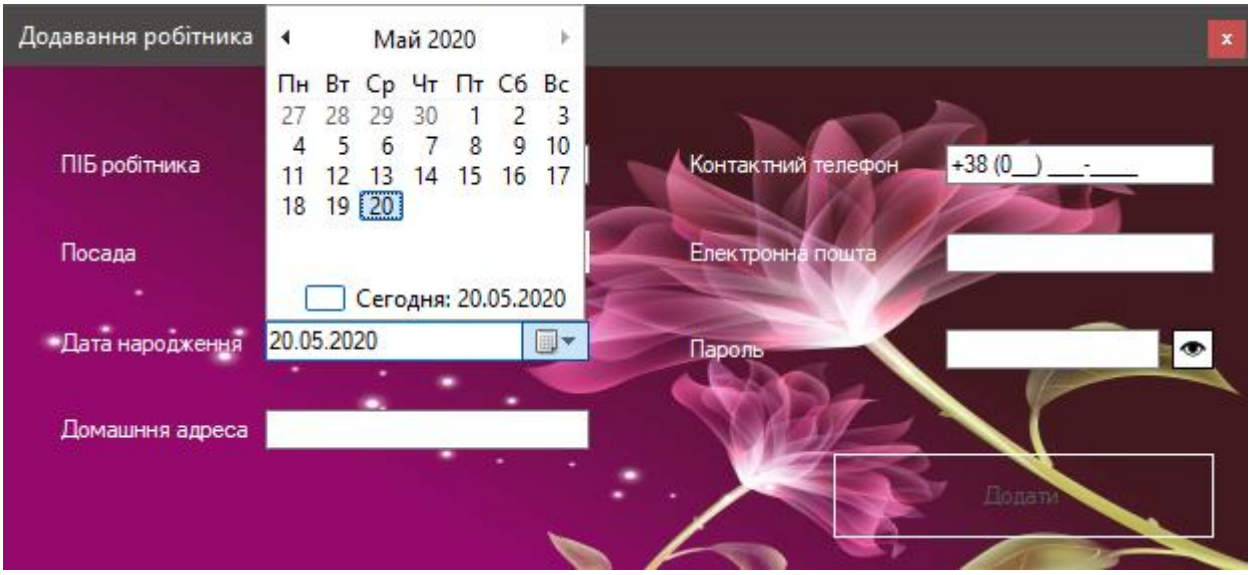


Рис. 2.13 — Використання DateTimePicker

Використав на майже на всіх формах обробник подій KeyPress для компонентів класу TextBox. Перевіряю вхідні дані. Наприклад, якщо користувачем був введен заборонений символ то він ігноруеться.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для користувача необхідно мати такі мінімальні параметри ЕОМ :

- ЦП [CPU]: Pentium 4.
- Відеоадаптер [GPU]: 3D адаптер nVidia, Intel, AMD/ATI.
- Відеопам’ять [VRAM]: 64 МБ.
- Накопічувач [HDD]: 120 МБ.
- Оперативна пам’ять [RAM]: 2048 МБ.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

– VisualStudio 2019 Community;

– SQLite Studio;

– Git, GitHub, GithubDesktop.

2.7.3. Виклик та завантаження програми

Для роботи з розробленим программним додатком потрібно запустити виконуючий файл (.exe) програми. Після чого потрібно авторизуватися, ввівши Номер робітника та пароль

2.7.4. Опис інтерфейсу користувача

При відкритті программи потрібно авторизуватися у програмі як адміністратор або менеджер, щоб увійти до додатку потрібоно ввести номер робітника та пароль, також можно продивитися пароль, щоб не зробити помилки, що зображено на рисунку 2.14.

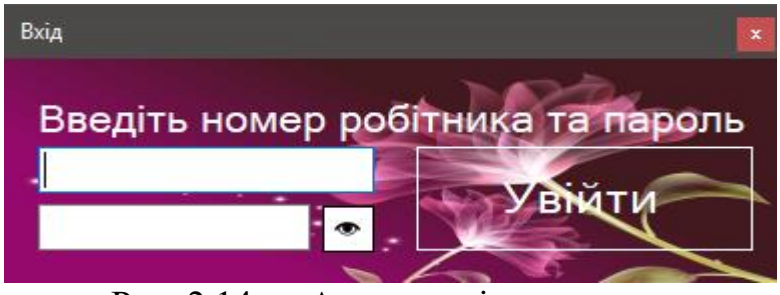


Рис. 2.14 — Авторизація у додатку

Якщо вход у програму був вдалий у нас відкриється форма, де можна буде побачити певну кількість вкладок. Де можна буде продивитися таблиці де

зберігається інформація та виконувати дії з ними. Наприклад, на першій вкладці «Рослини» ми бачемо інформацію про рослини і для зручного користування таблицею маємо фільтр де ми можемо вводити або обирати потрібні нам дані також забезпечена кнопка очищення фільтру, що зображено на рисунку 2.15.

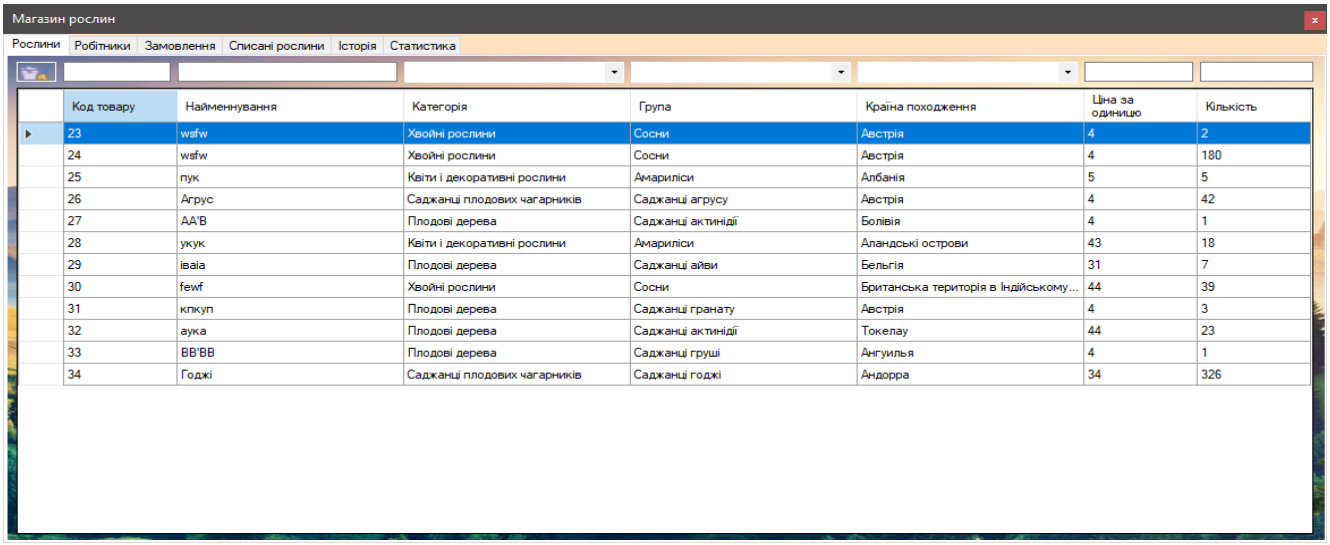


Рис. 2.15 — Головна форма

В залежності від посади робітника йому надаються різні права, тобто якщо посада робітника була «Адміністратор» то він може додати робітника який буде влаштовуватися на роботу, адміністратор обирає «Додати нового робітника» і вводе дані робітника, ввод ПІБ та адреси розрахований на те що адміністатор відвідально вводить дані нового робітника, що зображено на рисунку 2.16.

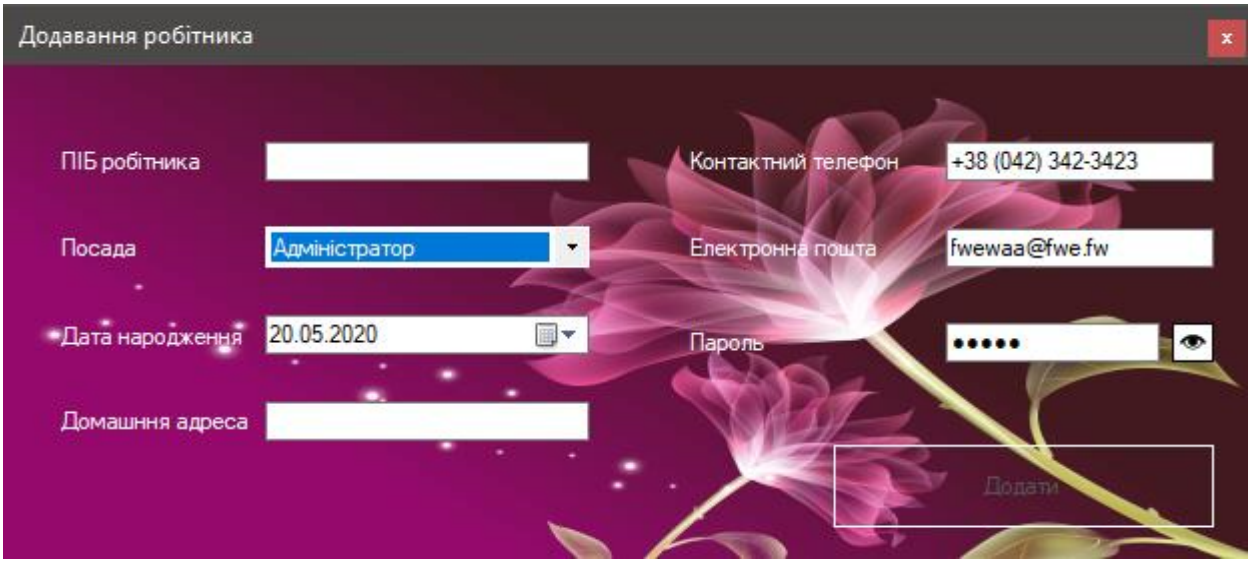


Рис. 2.16 — Додавання нового робітника

Також у можливостях адміністратора видаляти робітників по одному запису, тому при потребі обирається потрібний робітник, потім обирається пункт «Видалити робітника» і перевіряється чи впевнен у видалені, що хочете видалити робітника, що зображено на рисунку 2.17.

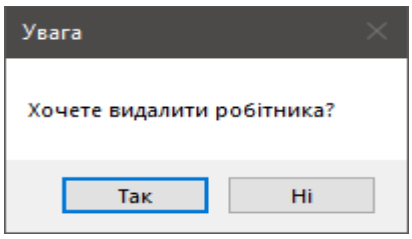


Рис. 2.17 — Видалення робітника

Можливо у будь-якого робітника, що зміняться дані, наприклад посада, яку може змінити тільки адміністратор тому він може редагувати інформацію менеджерів, а також менеджери можуть самі редагувати інформацію, але тільки особисту, окрім певних даних, таких як посада. Якщо вхід був здійснен адміністратором то йому потібно обрати робітника якого потрібно редагувати, обирає пункт «Редагувати дані», але якщо вхід був здійснен менеджером нема потреба обирати робітника, він сразу обирає пункт «Редагувати дані». Форма з редагування даних зображена на рисунку 2.18.

Рис. 2.18 — Редагування даних

Якщо виникла потреба у зміні паролю, обираємо пункт зміна паролю, відкривається форма де потрібно ввести старий пароль, а також новий і повторити

його, для того щоб було легче його ввести можно увімкнути функцію бачити пароль, а потім зберігти пароль, що зображено на рисунку 2.19.

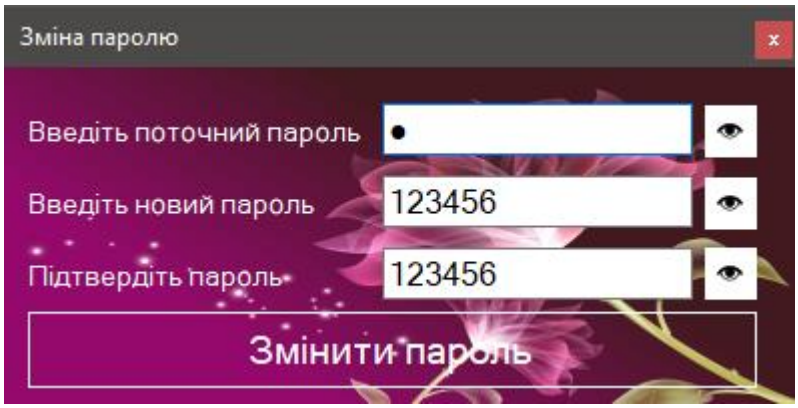


Рис. 2.19 — Зміна паролю

При завезені нових рослин потрібно відкрити форму «Додати нові рослини», але до обробки інформації з рослинами допускаються тільки менеджери, тобто адміністратори можуть тільки бачити список рослин, де ми пишемо назву рослини, обираємо її категорію та групу, до речі групу неможливо обрати до вибору категорії, також із списка обирається країна, де представлені усі країни світу і також вводемо ціну за одиницю та кількість рослин, що зображено на рисунку 2.20.

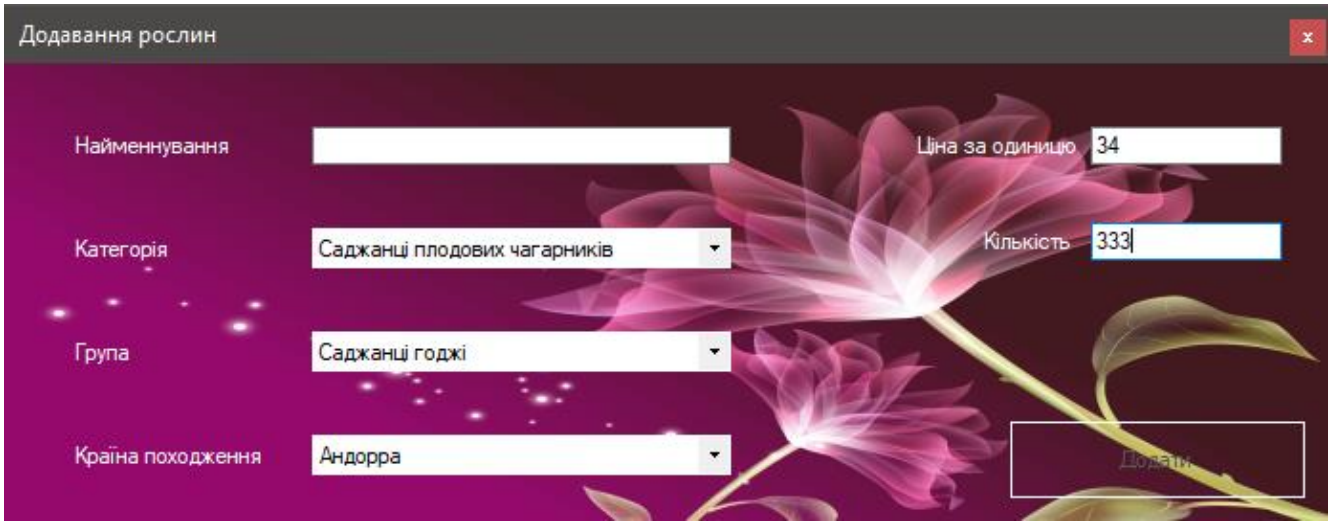


Рис. 2.20 — Додавання нових рослин

При необхідності редагувати рослини обираємо потрібну рослини і нажимаємо «Редагувати» і відкривається форма де і додавали рослини, окрім країни походження і потом нажимаємо редагувати, але при цьому усі поля повині бути заповнені, що зображено на рисунку 2.21.

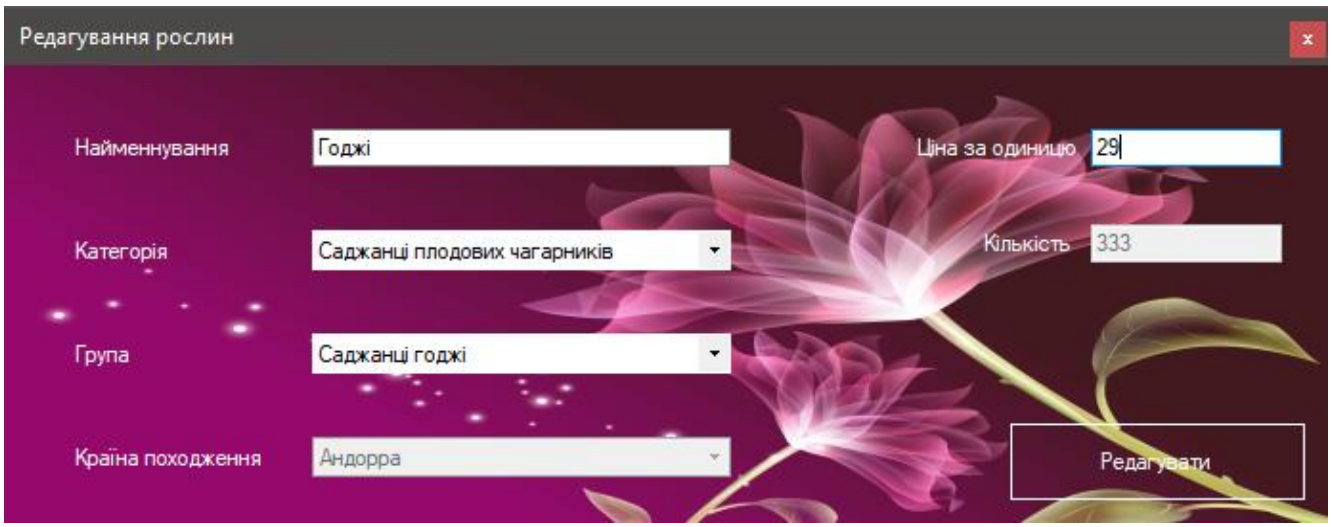


Рис. 2.21 — Редагування рослин

Якщо до магазину завезли нові рослини то щоб додати їх до потібно обрати рослину нажати кнопку «Додати до існуючих» і ввести потрібну кількість, що зображено на рисунку 2.22.

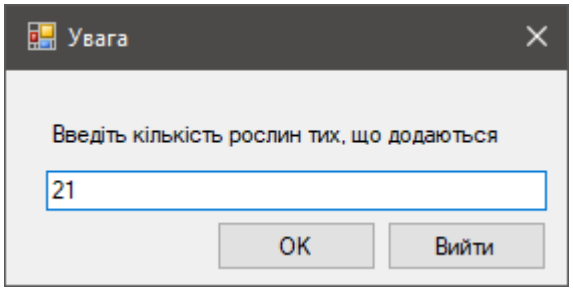


Рис. 2.22 — Додавання рослин до існуючих

Якщо менеджер помітив що рослини зів’яли йому потрібно їх списати, обирає потрібну рослину, обираємо пункт «Списати», вводе кількість рослин які зів’яли і із списку обирає причину списання, що зображено на рисунку 2.23.

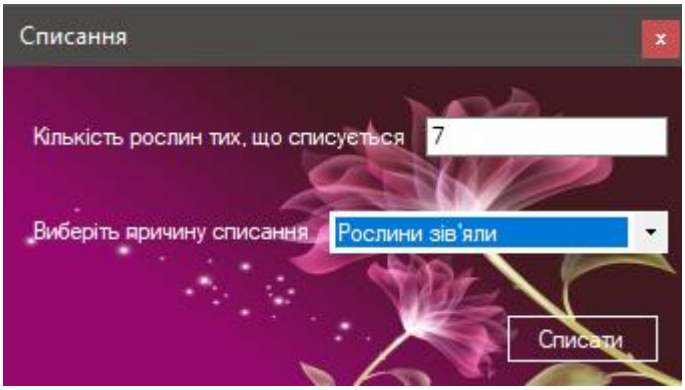


Рис. 2.23 — Списання рослин

При замовленні рослин обирається потрібна рослина, нажимаємо «Замовити», вводемо потрібну кількість і відкривається форма де офомляється замовлення при згоді клієнта і де ми можемо повністю побачити інформацію про дане замовлення, що зображено на рисунку 2.24.

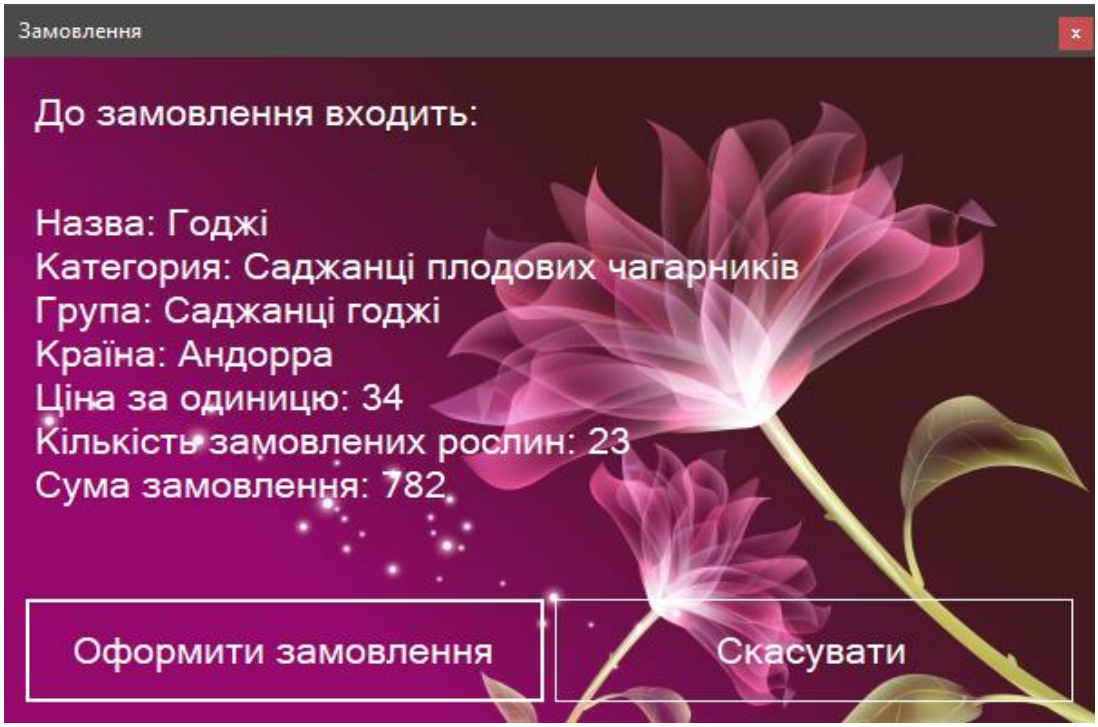
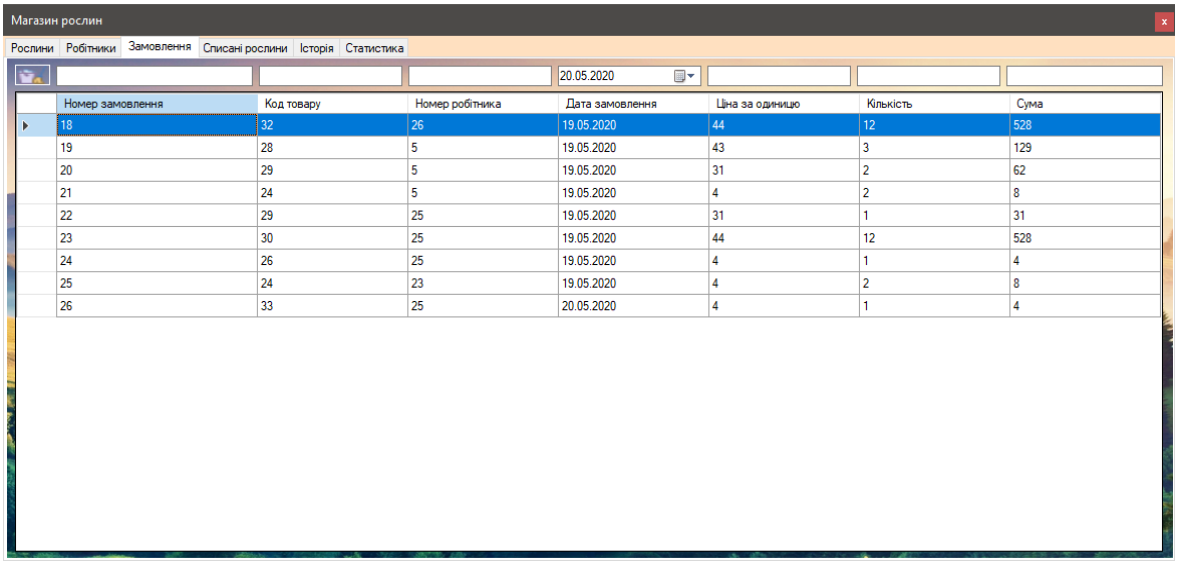


Рис. 2.24 — Оформлення замовлення

Також можна перейти на вкладку «Замовлення» які можуть продивитися усі робітники та фільтрувати записи. Там збережені замовленя зроблені усіма менеджерами за весь час, що зображено на рисунку 2.25.



Номер замовлення	Код товару	Номер робітника	Дата замовлення	Ціна за одиницю	Кількість	Сума
18	32	26	19.05.2020	44	12	528
19	28	5	19.05.2020	43	3	129
20	29	5	19.05.2020	31	2	62
21	24	5	19.05.2020	4	2	8
22	29	25	19.05.2020	31	1	31
23	30	25	19.05.2020	44	12	528
24	26	25	19.05.2020	4	1	4
25	24	23	19.05.2020	4	2	8
26	33	25	20.05.2020	4	1	4

Рис. 2.25 — Замовлення

Є вкладка «Історія» там збережені усі дії робітників за весь час і всіх робітників, але «Менеджер» може продивитися історію тільки своїх дій, а «Адміністратор» може продивитися дії усії робітників, та за допомогою фільтра вибрати певного робітника і переглянути його історію, що зображено на рисунку 2.26.

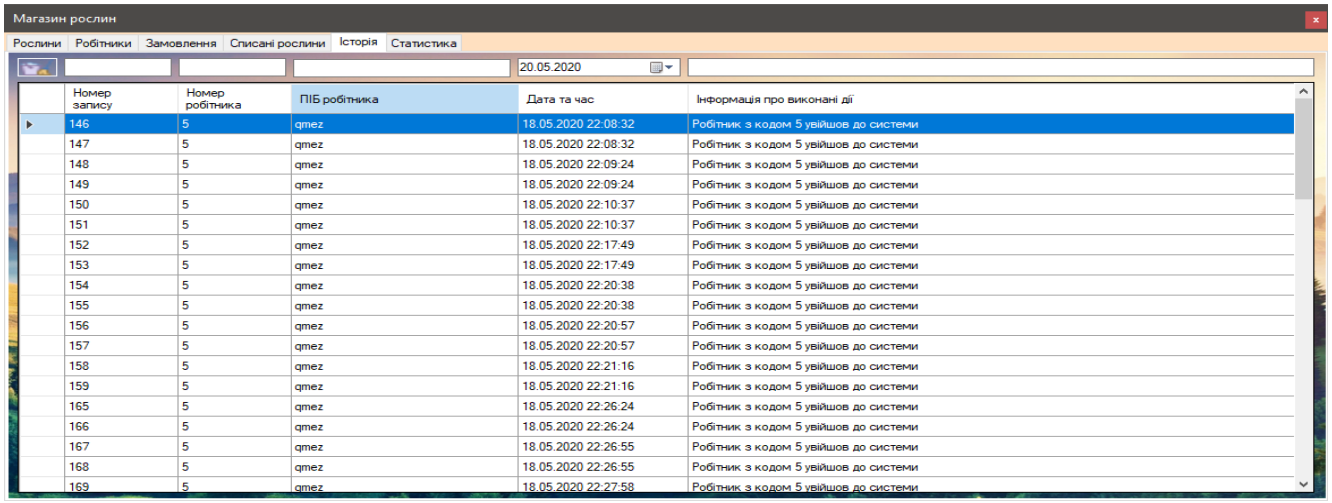


Рис. 2.26 — Історія

На вкладці «Статистика» можна побачити ще 2 вкладки на яких проходить прогноз на завтра в залежності від обраного періоду на основі зроблених замовлень, а також можна зробити прогноз на завтра в залежності від обраного періоду на основі зроблених замовлень певним менеджером, що зображено на рисунку 2.27.

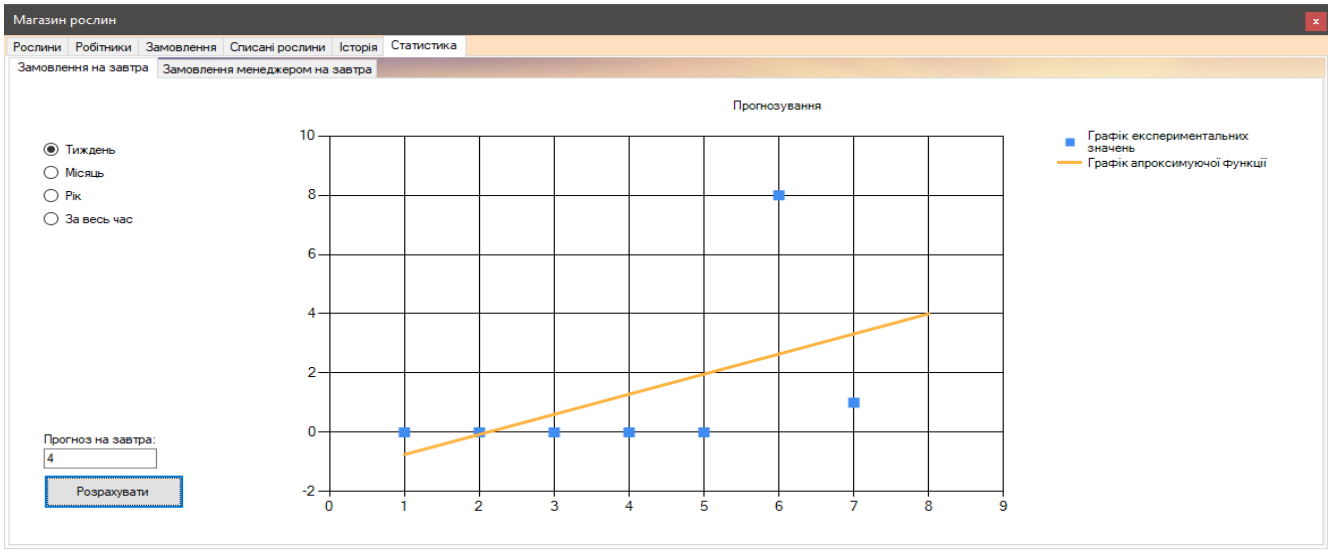


Рис. 2.27 — Історія

І на останнє коли менеджер оформлює замовлення він може роздрукувати чек за згодою клієнта, дані автоматично переносяться з програми до файлу з розширенням .xlsx, потім дані форматуються і починається автоматично друк без жодного натискання, це все виконується в фоновому режимі, результат перенесення даних зображено на рисунку 2.28.

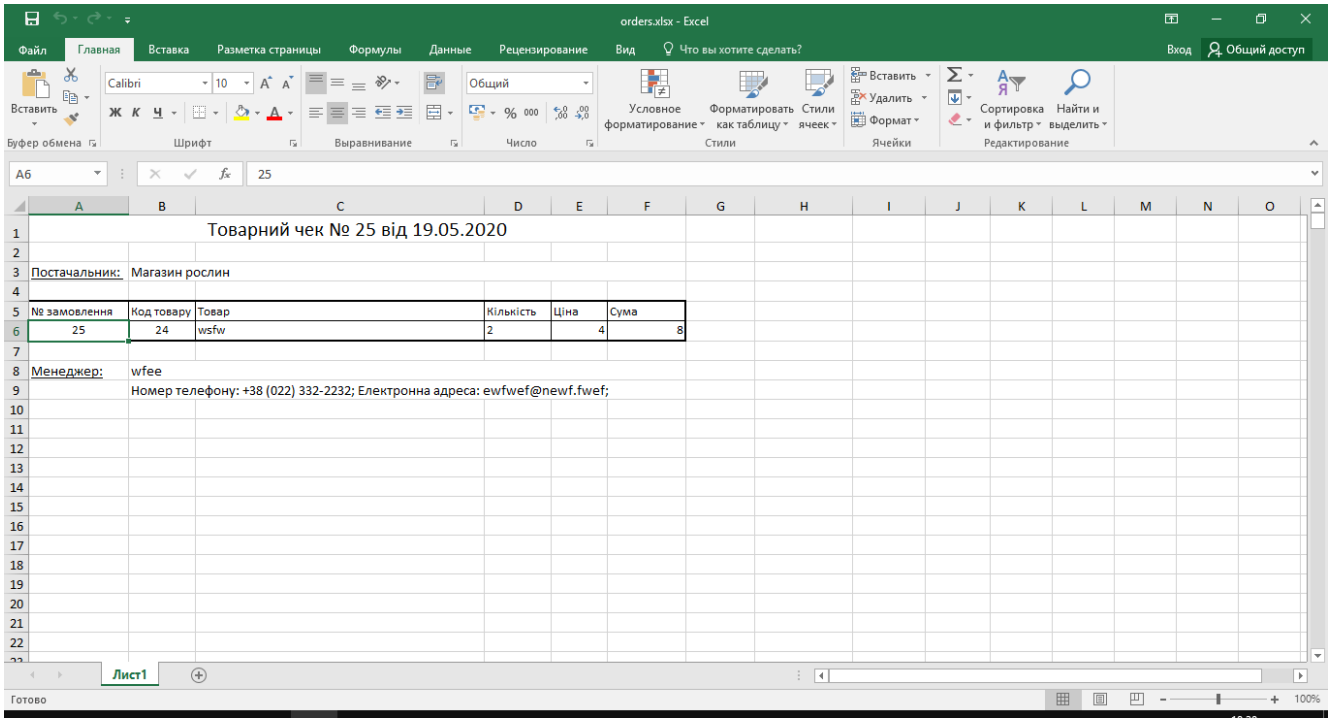


Рисунок 2.28 — Чек перенесений до файлу файлу з розширенням .xlsx

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів – 1000;

2. коефіцієнт складності програми – 1.5;

3. коефіцієнт корекції програми в ході її розробки – 0.09;

4. годинна заробітна плата програміста, грн/год – 78;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.2;

7. вартість машино-години, грн/год – 10.

Нормування праці в процесі створення програмного забезпечення істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки програмного забезпечення може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки програмного забезпечення можна розрахувати за формулою:

$$t = t_0 + t_u + t_a + t_n + t_{отл} + t_д, людино - годин, \quad (3.1)$$

де t_0 - витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування за готовою блок-схемою;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється. Умовне число операторів (тегів):

$$Q = q * C * (1 + p), \quad (3.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

Звідси умовне число операторів в програмі:

$$Q = 1000 * 1{,}5 * (1 + 0{,}09) = 1635. \quad (3.3)$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q*B}{(75...85)*k}, людино - годин, \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{1635*1{,}3}{85*1{,}2} = 21, людино - годин. \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, людино - годин \quad (3.6)$$

$$t_a = \frac{1635}{25*1,2} = 55, людино - годин. \quad (3.7)$$

Витрати на складання програми за готовою блок-схемою:

$$t_n = \frac{Q}{(20...25)*k}, людино - годин, \quad (3.8)$$

$$t_n = \frac{1635}{25*1,2} = 55, людино - годин. \quad (3.9)$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)*k}, людино - годин. \quad (3.10)$$

$$t_{отл} = \frac{1635}{5*1,2} = 273, людино - годин. \quad (3.11)$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1,5 * t_{отл}, людино - годин. \quad (3.12)$$

$$t_{отл}^{k} = 1,5 * 273 = 410, людино - годин. \quad (3.13)$$

Витрати праці на підготовку документації:

(3.14)

$$t_д = t_{др} + t_{до}, людино - годин,$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15...20*k}, людино - годин. \quad (3.15)$$

$$t_{др} = \frac{1635}{12*1,2} = 14, людино - годин, \quad (3.16)$$

де $t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 * t_{др}, людино - годин. \quad (3.17)$$

$$t_{до} = 0,75 * 14 = 11, людино - годин. \quad (3.18)$$

$$t_д = 14 + 11 = 25, людино - годин. \quad (3.19)$$

Тепер розрахуємо трудомісткість ПЗ:

$$t = 50 + 21 + 55 + 55 + 273 + 25 = 479, людино - годин. \quad (3.20)$$

3.2. Розрахунок витрат на створення програми

Витрати на створення програмного забезпечення Витрати на створення даного продукту $К_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{по} = З_{зп} + З_{мв}, грн. \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, грн, \quad (3.22)$$

де t - загальна трудомісткість, людино-годин;

$С_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 479 * 78 = 37,362грн. \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * С_{мч}, грн, \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год.

$$З_{мв} = 273 * 10 = 2730 грн. \quad (3.25)$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП:

$$К_{по} = 37,362 + 2730 = 40,092грн. \quad (3.26)$$

Очікуваний період створення ПЗ:

$$\mathrm{T} = \frac{\mathrm{t}}{\mathrm{B_k * F_p}}, \text{міс}, \qquad (3.27)$$

де B_k - число виконавців (1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$\mathrm{T} = \frac{479}{1*176} = 3 \text{ міс}. \qquad (3.28)$$

Висновки: У кваліфікаційній роботі була створена автоматизована платформа для бібліотечної системи. В третьому (економічному) розділі були визначені витрати на створення ПЗ, зазначеного у технічному завданні КР – 40,092 грн. і час створення програмного забезпечення – 3 місяці.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи була розроблена програма, що організовує систему взаємодії між клієнтами та працівниками приватної бізнес-установи з продажу квітів: покупцями та менеджерами, менеджерами та адміністраторами. Програмне забезпечення реалізує усі ланки бізнес установи, надаючи їм більш автоматизованого та оптимізованого функціоналу. Програма надає можливість вести облік реєстру рослин, робітників та контролювати процесси дій робітників, замовлень, списання рослин.

Розроблений додаток має наступні переваги:

– підтримує друк чеків, тобишь надійні відносини між клієнтами та магазином;

– зручний та зрозумілий інтерфейс, який забезпечує легку взаємодію співробітника з програмою, а, отже, й максимально швидке обслуговування;

– надійний захист інформації без ризику її втрати;

– універсальність та можливість підлаштовуватися під різні типи магазинів, зазнаючи мінімальних модифікацій.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (479 людино-годин), підраховані витрати на створення програмного забезпечення (40092 грн.) і гаданий період розробки (приблизно 3 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CloudShop [Електронний ресурс] — Режим доступу до ресурсу: https://cloudshop.ru/portal/2017/03/%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%B4%D0%BB%D1%8F-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0-%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2/.

2. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод / C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608с.

3. Кристиан Нейгел и др. C# 5.0 и платформа .NET 4.5 для профессионалов / Professional C# 5.0 and .NET 4.5. — М.: «Диалектика», 2013. — 1440с.

4. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание / C# Programming Language (Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784с.

5. Герберт Шилдт. C# 4.0: полное руководство / C# 4.0 The Complete Reference. — М.: «Вильямс», 2010. — С. 1056.

ДОДАТОК А

КОД ПРОГРАМИ

Program.cs

```
using System;
using System.Windows.Forms;

namespace FLOWERS
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("ro");
            MessageBoxManager.OK = "OK";
            MessageBoxManager.Cancel = "Вийти";
            MessageBoxManager.Yes = "Так";
            MessageBoxManager.No = "Ні";
            MessageBoxManager.Register();
            CSQLite lite = new CSQLite();
            Application.Run(new AUTORIZED());
            lite.close();
        }
    }
}
```

Autorized.cs

```
using System;
using System.Windows.Forms;

namespace FLOWERS
{
    public partial class AUTORIZED : Form
    {
        public AUTORIZED()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (CSQLite.password_employ(employ.Text, password.Text) == true)
            {
                CSQLite.open(employ.Text);
                Hide();
                MENU form1 = new MENU(employ.Text);
                form1.ShowDialog();
                Application.Exit();
            }
            else MessageBox.Show("Неправильно введені дані", "Увага");
```

```
        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (char.IsDigit(e.KeyChar) || (char)Keys.Back == e.KeyChar)
            {
                return;
            }
            e.Handled = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            password.UseSystemPasswordChar = password.UseSystemPasswordChar == false ? true
: false;
        }
    }
}
```

Menu.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace FLOWERS
{
    public partial class MENU : Form
    {
        string[] combo1 =
        {
            "Кипарисові",
            "Сосни",
            "Тис",
            "Туї",
            "Ялина",
            "Ялівець"
        };
        string[] combo2 =
        {
            "Саджанці абрикосу",
            "Саджанці азіміни",
            "Саджанці айви",
            "Саджанці актинідії",
            "Саджанці аличі",
            "Саджанці вишні",
            "Саджанці волоського горіха",
            "Саджанці горіха",
            "Саджанці гранату",
            "Саджанці груші",
            "Саджанці зизифуса",
            "Саджанці інжиру",
            "Саджанці ківі",
            "Саджанці колонних персиків",
            "Саджанці колонних слив",
            "Саджанці колоновидні абрикоса",
            "Саджанці колоновидною аличі",
            "Саджанці колоновидною вишні",
            "Саджанці колоновидною груші",
            "Саджанці лимона",
            "Саджанці мандарина",
            "Саджанці мигдалю",
            "Саджанці нектарина",
            "Саджанці пекан",
```

```
    "Саджанці персика",
    "Саджанці сливи",
    "Саджанці фісташки",
    "Саджанці фундука",
    "Саджанці хурми",
    "Саджанці черешні",
    "Саджанці шовковиці",
    "Саджанці яблунь"
};
string[] combo3 =
{
    "Алліум",
    "Амариліси",
    "Арум",
    "Барбарис",
    "Бегонія",
    "Бугенвіллія",
    "Бузок",
    "Вейгела",
    "Гіацинти",
    "Гладіолуси",
    "Гортензія",
    "Жимолость",
    "Жоржини",
    "Злаки і трави",
    "Іриси",
    "Кали",
    "Кореневища півоній",
    "Крокуси",
    "Листяні декоративні дерева",
    "Ліани",
    "Лілії",
    "Нарциси",
    "Півонії",
    "Розсада петунії",
    "Саджанці айстри",
    "Саджанці аквилегии",
    "Саджанці анемони",
    "Саджанці астильби",
    "Саджанці аюгі",
    "Саджанці бругмансія",
    "Саджанці гейхер",
    "Саджанці герані",
    "Саджанці гібіскуса",
    "Саджанці дельфиниума",
    "Саджанці деревію",
    "Саджанці клематиса",
    "Саджанці конвалії",
    "Саджанці купина",
    "Саджанці лілійників",
    "Саджанці півоній",
    "Саджанці плакучих і кулястих дерев",
    "Саджанці плюща",
    "Саджанці фіалок",
    "Саджанці флокси",
    "Саджанці хризантем",
    "Саджанці юки",
    "Спірея",
    "Тамарікс",
    "Троянди англійські",
    "Троянди карликові",
    "Троянди плетисті",
    "Троянди флорибунда",
    "Троянди чайно-гібридні",
    "Тюльпани",
```

```
            "Фрітіллярія",
            "Хости",
        };
        string[] combo4 =
        {
            "Вегетуючі саджанці винограду",
            "Кишмишні сорти винограду",
            "Саджанці агрусу",
            "Саджанці годжі",
            "Саджанці жимолості їстівної",
            "Саджанці журавлини",
            "Саджанці землеклубнікі",
            "Саджанці кизилу",
            "Саджанці кольоровий смородини",
            "Саджанці лохини",
            "Саджанці малини",
            "Саджанці обліпихи",
            "Саджанці ожини",
            "Саджанці полуниці",
            "Саджанці чорної смородини",
            "Столові сорти винограду"
        };
        int combo_index = -1;
        int combo_WRindex = -1;
        int SelCel = 0;
        int employMax = 0;
        bool flag = false;
        string ID_EMPLOY = null;
        DateTime date_today = DateTime.Now;
        string GetIdPlant() =>
grid_plant.Rows[grid_plant.CurrentCell.RowIndex].Cells[0].Value.ToString();
        string GetIdEmploy() =>
grid_employ.Rows[grid_employ.CurrentCell.RowIndex].Cells[0].Value.ToString();
        string GetIdEmployPrognoz() =>
grid_prognoz.Rows[grid_prognoz.CurrentCell.RowIndex].Cells[0].Value.ToString();
        public MENU(string id)
        {
            InitializeComponent();
            this.ID_EMPLOY = id;
            filter_comboBox();
            filter_WRcomboBox();
            if (flag) reader_grids();
            else reader_grids_history();
            if (CSQLite.GetAdmin(ID_EMPLOY) == "Адміністратор")
            {
                this.grid_employ.CellMouseDown += new
System.Windows.Forms.DataGridViewCellMouseEventHandler(this.mouse_rigth_employ);
                flag = true;
                contextMenuStrip1.Items.Clear();
            }
            else
            {
                flag = false;
                this.grid_employ.SelectionChanged += new
System.EventHandler(this.grid_employ_SelectionChanged);
                for (int i = 0; i < employMax; i++) if ((string)grid_employ[0, i].Value ==
id) SelCel = i;
                grid_employ.Rows[SelCel].Selected = true;
                contextMenuStrip2.Items.RemoveAt(3);
                contextMenuStrip2.Items.RemoveAt(2);
            }
            combo_group.Sorted = true;
            text_wr_group.Sorted = true;
            date.MaxDate = date_today;
            date_order.MaxDate = date_today;
```

```
            date_time.MaxDate = date_today;
            text_wr_date.MaxDate = date_today;
            this.grid_plant.CellMouseDown += new
System.Windows.Forms.DataGridViewCellMouseEventHandler(this.mouse_rigth_plant);
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void додатиНовіКвітиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ADDUPDATEPLANT aDDUPDATEPLANT = new ADDUPDATEPLANT(ID_EMPLOY);
            Hide();
            aDDUPDATEPLANT.ShowDialog();
            Show();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void додатиДоІснуючихToolStripMenuItem_Click(object sender, EventArgs e)
        {
            InputBox.InputBox inputBox = new InputBox.InputBox("Увага", "Введіть кількість
рослин тих, що додаються", true);
            CPlant cp = new CPlant();
            cp.id_plant = GetIdPlant();
            if ((cp.counter = inputBox.GetString()) == "")
            {
                MessageBox.Show("Нічого не введено", "Увага");
                return;
            }
            else CSQLite.sum_plant(cp, ID_EMPLOY);
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void редагуватиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ADDUPDATEPLANT aDDUPDATEPLANT = new ADDUPDATEPLANT(GetIdPlant(), ID_EMPLOY);
            Hide();
            aDDUPDATEPLANT.ShowDialog();
            Show();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void списатиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            WRITE_OFF wRITE_OFF = new WRITE_OFF(GetIdPlant(), ID_EMPLOY);
            Hide();
            wRITE_OFF.ShowDialog();
            Show();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void замовитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            InputBox.InputBox inputBox = new InputBox.InputBox("Увага", "Введіть кількість
замовляємих рослин", true);
            string str = inputBox.GetString();
            if (str == "" || str == null)
            {
                MessageBox.Show("Нічого не введено", "Увага");
                return;
            }
            ORDER oRDER = new ORDER(GetIdPlant(), str, ID_EMPLOY);
            oRDER.ShowDialog();
            Show();
```

```
    if (flag) reader_grids();
    else reader_grids_history();
}
private void button_clear_plant_Click(object sender, EventArgs e)
{
    clear_text_in_box_filter();
}
private void text_name_plant_TextChanged(object sender, EventArgs e)
{
    if (combo_index != 0)
        if (combo_categor.SelectedIndex == 0)
        {
            combo_group.Items.Clear();
            foreach (string arr in combo1) combo_group.Items.Add(arr);
        }
    if (combo_index != 1)
        if (combo_categor.SelectedIndex == 1)
        {
            combo_group.Items.Clear();
            foreach (string arr in combo2) combo_group.Items.Add(arr);
        }
    if (combo_index != 2)
        if (combo_categor.SelectedIndex == 2)
        {
            combo_group.Items.Clear();
            foreach (string arr in combo3) combo_group.Items.Add(arr);
        }
    if (combo_index != 3)
        if (combo_categor.SelectedIndex == 3)
        {
            combo_group.Items.Clear();
            foreach (string arr in combo4) combo_group.Items.Add(arr);
        }
    for (int i = 0; i < combo1.Length; i++)
        if (combo_group.Text == combo1[i])
        {
            combo_index = 0;
            combo_categor.SelectedIndex = 0;
            break;
        }
    for (int i = 0; i < combo2.Length; i++)
        if (combo_group.Text == combo2[i])
        {
            combo_index = 1;
            combo_categor.SelectedIndex = 1;
            break;
        }
    for (int i = 0; i < combo3.Length; i++)
        if (combo_group.Text == combo3[i])
        {
            combo_index = 2;
            combo_categor.SelectedIndex = 2;
            break;
        }
    for (int i = 0; i < combo4.Length; i++)
        if (combo_group.Text == combo4[i])
        {
            combo_index = 3;
            combo_categor.SelectedIndex = 3;
            break;
        }

    CPlant cp = new CPlant();
    cp.id_plant = text_id_plant.Text;
    cp.name = text_name_plant.Text;
```

```
            cp.categor = combo_categor.Text;
            cp.group_ = combo_group.Text;
            cp.country = combo_country.Text;
            cp.price = text_price.Text;
            cp.counter = text_count.Text;
            grid_plant.Rows.Clear();
            foreach (CPlant arr in CSQLite.filter_plant(cp))
                grid_plant.Rows.Add(arr.id_plant, arr.name, arr.categor, arr.group_,
arr.country, arr.price, arr.counter);
            combo_index = combo_categor.SelectedIndex;
        }
        private void button_clear_employ_Click(object sender, EventArgs e)
        {
            clear_text_in_box_filter();
        }
        private void text_address_TextChanged(object sender, EventArgs e)
        {
            date_time.MaxDate = DateTime.Now;
            CEmploy ce = new CEmploy();
            ce.phone = phone.MaskCompleted == true ? phone.Text : "";
            ce.id_employ = text_id_employ.Text;
            ce.full_name = text_full_name.Text;
            ce.position = combo_position.Text;
            ce.birthday = date.Value == date_today ? "" : date.Value.ToShortDateString();
            ce.address = text_address.Text;
            ce.mail = text_mail.Text;
            grid_employ.Rows.Clear();
            foreach (CEmploy arr in CSQLite.filter_employ(ce))
                grid_employ.Rows.Add(arr.id_employ, arr.full_name, arr.position,
arr.birthday, arr.address, arr.phone, arr.mail);
        }
        private void змінитиСвійПарольToolStripMenuItem_Click(object sender, EventArgs e)
        {
            EDITPASSWORD eDITPASSWORD = new EDITPASSWORD(ID_EMPLOY, ID_EMPLOY);
            Hide();
            eDITPASSWORD.ShowDialog();
            Show();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void видалитиРобітникаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Хочете видалити робітника?",
"Увага", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                CSQLite.delete_employ(GetIdEmploy(), ID_EMPLOY);
            }
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void редагуватиРобітникаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ADDUPDATEEMPLOY aDDUPDATEEMPLOY = new ADDUPDATEEMPLOY(GetIdEmploy(), ID_EMPLOY,
flag);
            Hide();
            aDDUPDATEEMPLOY.ShowDialog();
            Show();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void додатиНовогоРобітникаToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            ADDUPDATEEMPLOY aDDUPDATEEMPLOY = new ADDUPDATEEMPLOY(ID_EMPLOY);
```

```
    Hide();
    aDDUPDATEEMPLOY.ShowDialog();
    Show();
    if (flag) reader_grids();
    else reader_grids_history();
}
private void button_clear_write_off_Click(object sender, EventArgs e)
{
    clear_text_in_box_filter();
}
private void text_wr_categor_TextChanged(object sender, EventArgs e)
{
    if (combo_WRindex != 0)
        if (text_wr_categor.SelectedIndex == 0)
        {
            text_wr_group.Items.Clear();
            foreach (string arr in combo1) text_wr_group.Items.Add(arr);
        }
    if (combo_WRindex != 1)
        if (text_wr_categor.SelectedIndex == 1)
        {
            text_wr_group.Items.Clear();
            foreach (string arr in combo2) text_wr_group.Items.Add(arr);
        }
    if (combo_WRindex != 2)
        if (text_wr_categor.SelectedIndex == 2)
        {
            text_wr_group.Items.Clear();
            foreach (string arr in combo3) text_wr_group.Items.Add(arr);
        }
    if (combo_WRindex != 3)
        if (text_wr_categor.SelectedIndex == 3)
        {
            text_wr_group.Items.Clear();
            foreach (string arr in combo4) text_wr_group.Items.Add(arr);
        }
    for (int i = 0; i < combo1.Length; i++)
        if (text_wr_group.Text == combo1[i])
        {
            combo_WRindex = 0;
            text_wr_categor.SelectedIndex = 0;
            break;
        }
    for (int i = 0; i < combo2.Length; i++)
        if (text_wr_group.Text == combo2[i])
        {
            combo_WRindex = 1;
            text_wr_categor.SelectedIndex = 1;
            break;
        }
    for (int i = 0; i < combo3.Length; i++)
        if (text_wr_group.Text == combo3[i])
        {
            combo_WRindex = 2;
            text_wr_categor.SelectedIndex = 2;
            break;
        }
    for (int i = 0; i < combo4.Length; i++)
        if (text_wr_group.Text == combo4[i])
        {
            combo_WRindex = 3;
            text_wr_categor.SelectedIndex = 3;
            break;
        }
```

```
            date_time.MaxDate = DateTime.Now;
            CWrite_off cw = new CWrite_off();
            cw.id_plant = text_wr_id.Text;
            cw.name = text_wr_name.Text;
            cw.categor = text_wr_categor.Text;
            cw.group_ = text_wr_group.Text;
            cw.country = text_wr_country.Text;
            cw.counter = text_wr_count.Text;
            cw.date_ = text_wr_date.Value == date_today ? "" :
text_wr_date.Value.ToShortDateString();
            cw.reason = text_wr_reason.Text;
            grid_write_off.Rows.Clear();
            foreach (CWrite_off arr in CSQLite.filter_write_off(cw))
                grid_write_off.Rows.Add(arr.id_write_off, arr.id_plant, arr.name,
arr.categor, arr.group_, arr.country, arr.counter, arr.date_, arr.reason);
            combo_WRindex = text_wr_categor.SelectedIndex;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            chart1.Series[0].Points.Clear();
            chart1.Series[1].Points.Clear();
            int size, count = 0;
            DateTime date_today = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day);
            List<string> mas = CSQLite.prognoz_count_plant();
            List<double> prognoz = new List<double>();
            if (radio_prognoz1.Checked)
            {
                size = 6;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            else if (radio_prognoz2.Checked)
            {
                size = 29;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            else if (radio_prognoz3.Checked)
            {
                size = 364;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            else
            {
                DateTime date1 = new DateTime(2020, 01, 01); // год - месяц - день
                DateTime date2 = DateTime.Now;
                size = int.Parse(Math.Round((date2 - date1).TotalDays - 1).ToString());
                for (int j = size; j >= 0; j--)
```

```
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            List<double> X = new List<double>(0);
            for (int i = 0; i < size + 1; i++) X.Add(i + 1);
            text_prognoz.Text = Math.Round(MNK1(X, prognoz, size)).ToString();

        }
        private double MNK1(List<double> X, List<double> Y, double size)
        {
            double[] Y1 = new double[int.Parse(size.ToString()) + 2];
            double p1 = 0, p2 = 0, p3 = 0, p4 = 0;
            double a = 0, b = 0;
            for (int i = 0; i < X.Count; i++)
            {
                p1 += X[i] * Y[i];
                p2 += X[i];
                p3 += Y[i];
                p4 += X[i] * X[i];
            }
            a = ((X.Count * p1 - p2 * p3) / (Y.Count * p4 - p2 * p2));
            b = ((p3 * p4 - p1 * p2) / (X.Count * p4 - p2 * p2));
            X.Add(X.Count + 1);
            for (int i = 0; i < size + 2; i++) Y1[i] = a * X[i] + b;
            for (int i = 0; i < size + 2; i++)
            {
                if (i != size + 1) chart1.Series[0].Points.AddXY(X[i], Y[i]);
                chart1.Series[1].Points.AddXY(X[i], Y1[i]);
            }
            return a * (X.Count) + b;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            chart2.Series[0].Points.Clear();
            chart2.Series[1].Points.Clear();
            int size, count = 0;
            DateTime date_today = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day);
            List<string> mas = CSQLite.prognoz_count_plant_is_employ(GetIdEmployPrognoz());
            List<double> prognoz = new List<double>();
            if (radio_prognoz1_employ.Checked)
            {
                size = 6;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            else if (radio_prognoz2_employ.Checked)
            {
                size = 29;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
```

```
            }
            else if (radio_prognoz3_employ.Checked)
            {
                size = 364;
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            else
            {
                DateTime date1 = new DateTime(2020, 01, 01); // год - месяц - день
                DateTime date2 = DateTime.Now;
                size = int.Parse(Math.Round((date2 - date1).TotalDays - 1).ToString());
                for (int j = size; j >= 0; j--)
                {
                    for (int i = 0; i < mas.Count; i++) if (DateTime.Parse(mas[i]) ==
date_today.AddDays(-j)) count++;
                    prognoz.Add(double.Parse(count.ToString()));
                    count = 0;
                }
            }
            List<double> X = new List<double>(0);
            for (int i = 0; i < size + 1; i++) X.Add(i + 1);
            text_prognoz_employ.Text = Math.Round(MNK2(X, prognoz, size)).ToString();

        }
        private double MNK2(List<double> X, List<double> Y, double size)
        {
            double[] Y1 = new double[int.Parse(size.ToString()) + 2];
            double p1 = 0, p2 = 0, p3 = 0, p4 = 0;
            double a = 0, b = 0;
            for (int i = 0; i < X.Count; i++)
            {
                p1 += X[i] * Y[i];
                p2 += X[i];
                p3 += Y[i];
                p4 += X[i] * X[i];
            }
            a = ((X.Count * p1 - p2 * p3) / (Y.Count * p4 - p2 * p2));
            b = ((p3 * p4 - p1 * p2) / (X.Count * p4 - p2 * p2));
            X.Add(X.Count + 1);
            for (int i = 0; i < size + 2; i++) Y1[i] = a * X[i] + b;
            for (int i = 0; i < size + 2; i++)
            {
                if (i != size + 1) chart2.Series[0].Points.AddXY(X[i], Y[i]);
                chart2.Series[1].Points.AddXY(X[i], Y1[i]);
            }
            return a * (X.Count) + b;
        }
        private void button1_Click_1(object sender, EventArgs e)
        {
            clear_text_in_box_filter();
        }
        private void number_order_TextChanged(object sender, EventArgs e)
        {
            date_time.MaxDate = DateTime.Now;
            COrder co = new COrder();
            co.id_order = number_order.Text;
            co.id_plant = number_plant.Text;
            co.id_employ = numer_employ.Text;
```

```
            co.date_ = date_order.Value == date_today ? "" :
date_order.Value.ToShortDateString();
            co.price = price_order.Text;
            co.counter = count_order.Text;
            co.suma = sum_order.Text;
            grid_order.Rows.Clear();
            foreach (COrder arr in CSQLite.filter_order(co))
                grid_order.Rows.Add(arr.id_order, arr.id_plant, arr.id_employ, arr.date_,
arr.price, arr.counter, arr.suma);
        }
        private void text_record_TextChanged(object sender, EventArgs e)
        {
            date_time.MaxDate = DateTime.Now;
            CHistory ch = new CHistory();
            ch.id_record = text_id_record.Text;
            ch.id_employ = text_idemploy.Text;
            ch.date_time = date_time.Value == date_today ? "" :
date_time.Value.ToShortDateString();
            ch.record_history = text_record.Text;
            ch.name = text_name_hisory.Text;
            grid_history.Rows.Clear();
            foreach (CHistory arr in CSQLite.filter_history(ch))
                grid_history.Rows.Add(arr.id_record, arr.id_employ, arr.name, arr.date_time,
arr.record_history);
        }
        private void button_clear_history_Click(object sender, EventArgs e)
        {
            clear_text_in_box_filter();
        }
        void reader_grids()
        {
            grid_plant.Rows.Clear();
            grid_employ.Rows.Clear();
            grid_order.Rows.Clear();
            grid_history.Rows.Clear();
            grid_write_off.Rows.Clear();
            grid_prognoz.Rows.Clear();
            foreach (CPlant arr in CSQLite.view_plant())
                grid_plant.Rows.Add(arr.id_plant, arr.name, arr.categor, arr.group_,
arr.country, arr.price, arr.counter);
            foreach (CHistory arr in CSQLite.view_history())
                grid_history.Rows.Add(arr.id_record, arr.id_employ, arr.name, arr.date_time,
arr.record_history);
            foreach (CEmploy arr in CSQLite.view_employ())
                grid_employ.Rows.Add(arr.id_employ, arr.full_name, arr.position,
arr.birthday, arr.address, arr.phone, arr.mail);
            employMax = grid_employ.Rows.Count;
            grid_employ.Rows[SelCel].Selected = true;
            foreach (CWrite_off arr in CSQLite.view_write_off())
                grid_write_off.Rows.Add(arr.id_write_off, arr.id_plant, arr.name,
arr.categor, arr.group_, arr.country, arr.counter, arr.date_, arr.reason);
            foreach (COrder arr in CSQLite.view_order())
                grid_order.Rows.Add(arr.id_order, arr.id_plant, arr.id_employ, arr.date_,
arr.price, arr.counter, arr.suma);
            foreach (CEmploy arr in CSQLite.select_employ_meneger())
grid_prognoz.Rows.Add(arr.id_employ, arr.full_name);
        }
        void reader_grids_history()
        {
            grid_plant.Rows.Clear();
            grid_employ.Rows.Clear();
            grid_order.Rows.Clear();
            grid_history.Rows.Clear();
            grid_write_off.Rows.Clear();
            grid_prognoz.Rows.Clear();
```

```
            foreach (CPlant arr in CSQLite.view_plant())
                grid_plant.Rows.Add(arr.id_plant, arr.name, arr.categor, arr.group_,
arr.country, arr.price, arr.counter);
            foreach (CHistory arr in CSQLite.view_history_admin(ID_EMPLOY))
                grid_history.Rows.Add(arr.id_record, arr.id_employ, arr.name, arr.date_time,
arr.record_history);
            foreach (CEmploy arr in CSQLite.view_employ())
                grid_employ.Rows.Add(arr.id_employ, arr.full_name, arr.position,
arr.birthday, arr.address, arr.phone, arr.mail);
            employMax = grid_employ.Rows.Count;
            grid_employ.Rows[SelCel].Selected = true;
            foreach (CWrite_off arr in CSQLite.view_write_off())
                grid_write_off.Rows.Add(arr.id_write_off, arr.id_plant, arr.name,
arr.categor, arr.group_, arr.country, arr.counter, arr.date_, arr.reason);
            foreach (COrder arr in CSQLite.view_order())
                grid_order.Rows.Add(arr.id_order, arr.id_plant, arr.id_employ, arr.date_,
arr.price, arr.counter, arr.suma);
            foreach (CEmploy arr in CSQLite.select_employ_meneger())
grid_prognoz.Rows.Add(arr.id_employ, arr.full_name);
        }
        void filter_comboBox()
        {
            combo_group.Items.Clear();
            foreach (string arr in combo1) combo_group.Items.Add(arr);
            foreach (string arr in combo2) combo_group.Items.Add(arr);
            foreach (string arr in combo3) combo_group.Items.Add(arr);
            foreach (string arr in combo4) combo_group.Items.Add(arr);
        }
        void filter_WRcomboBox()
        {
            text_wr_group.Items.Clear();
            foreach (string arr in combo1) text_wr_group.Items.Add(arr);
            foreach (string arr in combo2) text_wr_group.Items.Add(arr);
            foreach (string arr in combo3) text_wr_group.Items.Add(arr);
            foreach (string arr in combo4) text_wr_group.Items.Add(arr);
        }
        private void clear_text_in_box_filter()
        {
            text_id_plant.Text = text_name_plant.Text = text_price.Text = text_count.Text =
"";
            combo_categor.SelectedIndex = combo_country.SelectedIndex =
combo_group.SelectedIndex = -1;
            text_id_employ.Text = text_address.Text = text_full_name.Text = text_mail.Text =
"";
            combo_position.SelectedIndex = -1;
            date.Value = date_today;
            number_order.Text = number_plant.Text = numer_employ.Text = price_order.Text =
count_order.Text = sum_order.Text = "";
            date_order.Value = date_today;
            textBox1.Text = textBox2.Text = "";
            text_wr_id.Text = text_wr_name.Text = text_wr_count.Text = "";
            text_wr_categor.SelectedIndex = text_wr_group.SelectedIndex =
text_wr_country.SelectedIndex = text_wr_reason.SelectedIndex = -1;
            text_wr_date.Value = date_today;
            text_id_record.Text = text_record.Text = text_idemploy.Text =
text_name_hisory.Text = "";
            date_time.MaxDate = DateTime.Now;
            date_time.Value = date_today;
            phone.Clear();
            filter_comboBox();
            filter_WRcomboBox();
            if (flag) reader_grids();
            else reader_grids_history();
        }
        private void MENU_FormClosing(object sender, FormClosingEventArgs e)
```

```
        {
            CSQLite.exit(ID_EMPLOY);

        }
        private void grid_employ_SelectionChanged(object sender, EventArgs e)
        {
            if (grid_employ.Rows.Count == employMax)
            {
                grid_employ.Rows[SelCel].Selected = true;
                grid_employ.CurrentCell = grid_employ.Rows[SelCel].Cells[0];
            }
        }
        private void mouse_rigth_plant (object sender, DataGridViewCellMouseEventArgs e)
        {
            if (e.Button == System.Windows.Forms.MouseButtons.Right && e.RowIndex >= 0 &&
e.ColumnIndex >= 0)
            {
                grid_plant.ClearSelection();
                grid_plant.Rows[e.RowIndex].Cells[e.ColumnIndex].Selected = true;
                grid_plant.CurrentCell = grid_plant.Rows[e.RowIndex].Cells[e.ColumnIndex];
            }
        }
        private void mouse_rigth_employ(object sender, DataGridViewCellMouseEventArgs e)
        {
            if (e.Button == System.Windows.Forms.MouseButtons.Right && e.RowIndex >= 0 &&
e.ColumnIndex >= 0)
            {
                grid_employ.ClearSelection();
                grid_employ.Rows[e.RowIndex].Cells[e.ColumnIndex].Selected = true;
                grid_employ.CurrentCell = grid_employ.Rows[e.RowIndex].Cells[e.ColumnIndex];
            }
        }
        private void clear_prognoz_Click(object sender, EventArgs e)
        {
            clear_text_in_box_filter();
        }
        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            grid_prognoz.Rows.Clear();
            foreach (CEmploy arr in CSQLite.filter_prognoz(textBox2.Text, textBox1.Text))
                grid_prognoz.Rows.Add(arr.id_employ, arr.full_name);
        }
    }
}
        [HttpPut]
        public IActionResult Update(UpdateAuthorModel updateAuthorModel)
        {
            try
            {
                var author =
_mapper.Map<BussinesLogic.Models.Authors.UpdateAuthorModel>(updateAuthorModel);
                authorService.Update(author);

                return Ok();
            }
            catch (Exception e)
            {
                return BadRequest(e.Message);
            }
        }
    }
}
```

Order.cs

```
using System;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;
using System.Reflection;
using System.IO;

namespace FLOWERS
{
    public partial class ORDER : Form
    {
        CPlant cp;
        COrder co = new COrder();
        Assembly assem = typeof(Program).Assembly;
        string excel_should = null;
        string ID_plant = null;
        string ID_employ = null;
        string counter = null;
        public ORDER(string id_plant, string count, string id_employ)
        {
            InitializeComponent();
            excel_should = (assem.Location.Substring(0, assem.Location.Length - 11)) +
"orders.xlsx";
            ID_plant = id_plant;
            ID_employ = id_employ;
            counter = count;
            cp = CSQLite.view_plant_select(ID_plant);
            co.id_plant = ID_plant;
            co.id_employ = ID_employ;
            co.date_ = DateTime.Now.ToShortDateString();
            co.price = cp.price;
            co.counter = counter;
            co.suma = (double.Parse(cp.price) * long.Parse(counter)).ToString();
            label_order.Text = "Назва: " + cp.name + Environment.NewLine + "Категория: " +
cp.categor + Environment.NewLine +
                "Група: " + cp.group_ + Environment.NewLine + "Країна: " + cp.country +
Environment.NewLine +
                "Ціна за одиницю: " + cp.price + Environment.NewLine + "Кількість замовлених
рослин: " + counter + Environment.NewLine +
                "Сума замовлення: " + double.Parse(cp.price) * double.Parse(counter);
        }
        private void button_cancel_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void button_order_Click(object sender, EventArgs e)
        {
            if (CSQLite.minus_plant(co, ID_employ))
            {
                DialogResult dialogResult = MessageBox.Show("Роздрукувати товарний чек?" +
Environment.NewLine + "Це займе деякий час", "Увага", MessageBoxButtons.YesNo);
                if (dialogResult == DialogResult.Yes)
                {
                    var excelApp = new Excel.Application();
                    excelApp.Visible = false;
                    var object_excel = excelApp.Workbooks.Open(excel_should);
                    Excel.Worksheet workSheet = excelApp.ActiveSheet;
                    CPlant cp = CSQLite.view_plant_select(co.id_plant);
                    CEmploy ce = CSQLite.view_employ_select(co.id_employ);
                    COrder order = CSQLite.view_order_select();

                    workSheet.Cells[1, "A"] = "Товарний чек № " + order.id_order + " від " +
order.date_;
                    workSheet.Cells[6, "A"] = order.id_order;
                    workSheet.Cells[6, "B"] = order.id_plant;
```

```
                    workSheet.Cells[6, "C"] = cp.name;
                    workSheet.Cells[6, "D"] = order.counter;
                    workSheet.Cells[6, "E"] = cp.price;
                    workSheet.Cells[8, "B"] = ce.full_name;
                    workSheet.Cells[9, "B"] = "Номер телефону: " + ce.phone + "; Електронна
адреса: " + ce.mail + ";";

                    try
                    {
                        //печать
                        workSheet.PrintOutEx();
                    }
                    catch { MessageBox.Show("Друк неможливий. Проблеми з принтером ", "
Увага"); }
                    object_excel.Save();
                    excelApp.Quit();
                }
                Close();
            }
        }
    }
}
```

Write_Off.cs

```
using System;
using System.Windows.Forms;
namespace FLOWERS
{
    public partial class WRITE_OFF : Form
    {
        string ID = null;
        string ID_EMPLOY = null;
        public WRITE_OFF(string id, string id_employ)
        {
            ID = id;
            ID_EMPLOY = id_employ;
            InitializeComponent();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && comboBox1.SelectedIndex != -1) button1.Enabled =
true;
            else button1.Enabled = false;
        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar <= 47 || e.KeyChar >= 58) && e.KeyChar != 8) e.Handled = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            CPlant cp = new CPlant();
            cp.id_plant = ID;
            cp.counter = textBox1.Text;
            if(CSQLite.minus_plant(cp, comboBox1.SelectedItem.ToString(), ID_EMPLOY))
Close();
            textBox1.Text = "";
            comboBox1.SelectedIndex = -1;
            MessageBox.Show("Списання було успішно виконано", "Увага");
        }
    }
}
```

ADDUPDATPLANT.CS

```
using System;
using System.Windows.Forms;

namespace FLOWERS
{

    public partial class ADDUPDATEPLANT : Form
    {
        int count = 0;
        string ID;
        string ID_EMPLOY;
        public ADDUPDATEPLANT(string id_employ)
        {
            InitializeComponent();
            ID_EMPLOY = id_employ;
            button_update.Visible = false;
            button_add.Visible = true;
            Text = "Додавання рослин";
        }
        public ADDUPDATEPLANT(string id_plant, string id_employ)
        {
            InitializeComponent();
            ID_EMPLOY = id_employ;
            button_update.Visible = true;
            button_add.Visible = false;
            Text = "Редагування рослин";
            ID = id_plant;
            CPlant cp = CSQLite.view_plant_select(ID);
            text_count.Enabled = false;
            combo_country.Enabled = false;
            text_name.Text = cp.name;
            text_price.Text = cp.price;
            text_count.Text = cp.counter;
            combo_categor.Text = cp.categor;
            combo_group.Text = cp.group_;
            combo_country.Text = cp.country;
        }

        string[] combo1 =
        {
            "Кипарисові",
            "Сосни",
            "Тис",
            "Туї",
            "Ялина",
            "Ялівець"
        };
        string[] combo2 =
        {
            "Саджанці абрикосу",
            "Саджанці азіміни",
            "Саджанці айви",
            "Саджанці актинідії",
            "Саджанці аличі",
            "Саджанці вишні",
            "Саджанці волоського горіха",
            "Саджанці горіха",
            "Саджанці гранату",
            "Саджанці груші",
            "Саджанці зизифуса",
            "Саджанці інжиру",
            "Саджанці ківі",
            "Саджанці колонних персиків",
```

```
    "Саджанці колонних слив",
    "Саджанці колоновидні абрикоса",
    "Саджанці колоновидною аличі",
    "Саджанці колоновидною вишні",
    "Саджанці колоновидною груші",
    "Саджанці лимона",
    "Саджанці мандарина",
    "Саджанці мигдалю",
    "Саджанці нектарина",
    "Саджанці пекан",
    "Саджанці персика",
    "Саджанці сливи",
    "Саджанці фісташки",
    "Саджанці фундука",
    "Саджанці хурми",
    "Саджанці черешні",
    "Саджанці шовковиці",
    "Саджанці яблунь"
};
string[] combo3 =
{
    "Алліум",
    "Амариліси",
    "Арум",
    "Барбарис",
    "Бегонія",
    "Бугенвіллія",
    "Бузок",
    "Вейгела",
    "Гіацинти",
    "Гладіолуси",
    "Гортензія",
    "Жимолость",
    "Жоржини",
    "Злаки і трави",
    "Іриси",
    "Кали",
    "Кореневища півоній",
    "Крокуси",
    "Листяні декоративні дерева",
    "Ліани",
    "Лілії",
    "Нарциси",
    "Півонії",
    "Розсада петунії",
    "Саджанці айстри",
    "Саджанці аквилегии",
    "Саджанці анемони",
    "Саджанці астильби",
    "Саджанці аюгі",
    "Саджанці бругмансія",
    "Саджанці гейхер",
    "Саджанці герані",
    "Саджанці гібіскуса",
    "Саджанці дельфиниума",
    "Саджанці деревію",
    "Саджанці клематиса",
    "Саджанці конвалії",
    "Саджанці купина",
    "Саджанці лілійників",
    "Саджанці півоній",
    "Саджанці плакучих і кулястих дерев",
    "Саджанці плюща",
    "Саджанці фіалок",
    "Саджанці флокси",
```

```
            "Саджанці хризантем",
            "Саджанці юки",
            "Спірея",
            "Тамарікс",
            "Троянди англійські",
            "Троянди карликові",
            "Троянди плетисті",
            "Троянди флорибунда",
            "Троянди чайно-гібридні",
            "Тюльпани",
            "Фрітіллярія",
            "Хости",
        };
        string[] combo4 =
        {
            "Вегетуючі саджанці винограду",
            "Кишмишні сорти винограду",
            "Саджанці агрусу",
            "Саджанці годжі",
            "Саджанці жимолості їстівної",
            "Саджанці журавлини",
            "Саджанці землеклубнікі",
            "Саджанці кизилу",
            "Саджанці кольоровий смородини",
            "Саджанці лохини",
            "Саджанці малини",
            "Саджанці обліпихи",
            "Саджанці ожини",
            "Саджанці полуниці",
            "Саджанці чорної смородини",
            "Столові сорти винограду"
        };

        private void combo_categor_SelectedIndexChanged(object sender, EventArgs e)
        {
            combo_group.Items.Clear();
            if (combo_categor.SelectedIndex == 0)
            {
                foreach (string arr in combo1) combo_group.Items.Add(arr);
            }
            if (combo_categor.SelectedIndex == 1)
            {
                foreach (string arr in combo2) combo_group.Items.Add(arr);
            }
            if (combo_categor.SelectedIndex == 2)
            {
                foreach (string arr in combo3) combo_group.Items.Add(arr);
            }
            if (combo_categor.SelectedIndex == 3)
            {
                foreach (string arr in combo4) combo_group.Items.Add(arr);
            }
            combo_group.Enabled = true;
        }

        private void text_price_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == '.') e.KeyChar = ',';
            if (char.IsDigit(e.KeyChar) || (char)Keys.Back == e.KeyChar || (e.KeyChar == ','
&& text_price.TextLength != 0 && text_price.TextLength < text_price.MaxLength - 1))
            {
                if (e.KeyChar == ',')
                {
                    for (int i = 0; i < text_price.TextLength; i++) if (text_price.Text[i]
== ',') { e.Handled = true; return; }
```

```
                }
                if (count == 2 && char.IsDigit(e.KeyChar)) { e.Handled = true; return; }
                if (text_price.TextLength > 1)
                {
                    if (text_price.Text[text_price.TextLength - 1 - count] == ',' &&
!(e.KeyChar == (char)Keys.Back)) count++;
                    else if (e.KeyChar == (char)Keys.Back && count > 0) count--;
                }
            }
            else e.Handled = true;
        }

        private void text_count_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (char.IsDigit(e.KeyChar) || (char)Keys.Back == e.KeyChar)
            {
                return;
            }
            e.Handled = true;
        }

        private void text_name_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar >= 33 && e.KeyChar <= 38) || (e.KeyChar >= 40 && e.KeyChar <= 44)
|| (e.KeyChar >= 46 && e.KeyChar <= 64) ||
                (e.KeyChar >= 91 && e.KeyChar <= 96) || (e.KeyChar >= 123 && e.KeyChar <=
127)) e.Handled = true;
        }

        private void text_name_TextChanged(object sender, EventArgs e)
        {
            if (text_name.Text != "" && text_count.Text != "" && text_price.Text != "" &&
                combo_categor.SelectedIndex != -1 && combo_country.SelectedIndex != -1 &&
combo_group.SelectedIndex != -1)
                button_add.Enabled = button_update.Enabled = true;
            else button_add.Enabled = button_update.Enabled = false;
        }

        private void button_add_Click(object sender, EventArgs e)
        {
            CPlant cp = new CPlant();
            cp.name = text_name.Text;
            cp.categor = combo_categor.Text;
            cp.group_ = combo_group.Text;
            cp.country = combo_country.Text;
            cp.price = text_price.Text;
            cp.counter = text_count.Text;
            CSQLite.insert_plant(cp, ID_EMPLOY);
            MessageBox.Show("Запис був успішно доданий", "Увага");
            text_name.Text = text_price.Text = text_count.Text = "";
            combo_categor.SelectedIndex = combo_group.SelectedIndex =
combo_country.SelectedIndex = -1;
            button_add.Enabled = button_update.Enabled = false;
        }

        private void button_update_Click(object sender, EventArgs e)
        {
            CPlant cp = new CPlant();
            cp.id_plant = ID;
            cp.name = text_name.Text;
            cp.categor = combo_categor.Text;
            cp.group_ = combo_group.Text;
            cp.country = combo_country.Text;
            cp.price = text_price.Text;
            cp.counter = text_count.Text;
```

```
            CSQLite.update_plant(cp, ID_EMPLOY);
            MessageBox.Show("Запис був успішно відредагований", "Увага");
            text_name.Text = text_price.Text = text_count.Text = "";
            combo_categor.SelectedIndex = combo_group.SelectedIndex =
combo_country.SelectedIndex = -1;
            button_add.Enabled = button_update.Enabled = false;
            Close();
        }
    }
}
```

ADDPDATEEmploy.cs

```
using System;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace FLOWERS
{
    public partial class ADDUPDATEEMPLOY : Form
    {
        string ID = null;
        string ID_EMPLOY;
        string ph, em;
        DateTime date_today = DateTime.Now;
        ToolTip tooltip = new ToolTip();
        public ADDUPDATEEMPLOY(string id_employ)
        {
            InitializeComponent();
            ID_EMPLOY = id_employ;
            date.MaxDate = date_today;
            button_update.Visible = false;
            button_add.Visible = true;
            Text = "Додавання робітника";
        }

        public ADDUPDATEEMPLOY(string id, string id_employ, bool admin)
        {
            ID = id;
            ID_EMPLOY = id_employ;
            InitializeComponent();
            button_update.Visible = true;
            button_add.Visible = false;
            Text = "Редагування робітника";
            date.MaxDate = date_today;
            date.Enabled = password.Enabled = button_watch_password.Enabled = false;
            CEmploy ce = CSQLite.view_employ_select(ID);
            password.Text = ce.password;
            name.Text = ce.full_name;
            position.Text = ce.position;
            if (admin && ID_EMPLOY != id) position.Enabled = true;
            else position.Enabled = false;
            date.Value = DateTime.ParseExact(ce.birthday, "dd.MM.yyyy", null);
            adres.Text = ce.address;
            phone.Text = ce.phone;
            mail.Text = ce.mail;
            ph = ce.phone;
            em = ce.mail;
        }

        private void button_update_Click(object sender, EventArgs e)
        {
            CEmploy ce = new CEmploy();
```

```
            ce.id_employ = ID;
            ce.password = password.Text;
            ce.full_name = name.Text;
            ce.position = position.Text;
            ce.birthday = date.Text;
            ce.address = adres.Text;
            ce.phone = phone.Text;
            ce.mail = mail.Text;
            if (!CSQLite.update_employ(ph, em, ce, ID_EMPLOY)) return;
            MessageBox.Show("Запис був успішно відредагован", "Увага");
            name.Text = mail.Text = date.Text = password.Text = adres.Text = "";
            phone.Clear();
            position.SelectedIndex = -1;
            button_add.Enabled = button_update.Enabled = false;
            Close();
        }

        private void button_add_Click(object sender, EventArgs e)
        {
            CEmploy ce = new CEmploy();
            ce.password = password.Text;
            ce.full_name = name.Text;
            ce.position = position.Text;
            ce.birthday = date.Text;
            ce.address = adres.Text;
            ce.phone = phone.Text;
            ce.mail = mail.Text;
            if (!CSQLite.insert_employ(ce, ID_EMPLOY)) return;
            MessageBox.Show("Запис був успішно додан", "Увага");
            Close();
        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {
            string pattern = @"^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";
            Match isMatch = Regex.Match(mail.Text, pattern, RegexOptions.IgnoreCase);
            if (name.Text != "" && mail.Text != "" && phone.MaskCompleted &&
                position.SelectedIndex != -1 && password.Text != "" && date.Text != "" &&
adres.Text != "" && isMatch.Success)
                button_add.Enabled = button_update.Enabled = true;
            else button_add.Enabled = button_update.Enabled = false;
        }

        private void button_watch_password_Click(object sender, EventArgs e) =>
password.UseSystemPasswordChar = password.UseSystemPasswordChar == false ? true : false;

        private void name_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar >= 33 && e.KeyChar <= 38) || (e.KeyChar >= 40 && e.KeyChar <= 44)
|| (e.KeyChar >= 46 && e.KeyChar <= 64) ||
                (e.KeyChar >= 91 && e.KeyChar <= 96) || (e.KeyChar >= 123 && e.KeyChar <=
127)) e.Handled = true;
        }

        private void mail_MouseEnter(object sender, EventArgs e)
        {
            tooltip.SetToolTip(mail, "Тільки латинські букви нижнього регістра, цифри,
крапка, тире та нижнє підкреслення" + Environment.NewLine +
                "Крапка, тире, нижнє підкреслення не може бути першою і останньою, вона не
може йти в слід за іншою точкою, тире, нижнім підкресленням");
        }
    }
}
```

SQLite.cs

```
using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Windows.Forms;

namespace FLOWERS
{
    class CSQLite
    {
        static SQLiteDataReader read;
        static SQLiteCommand command;
        static SQLiteConnection conn = new SQLiteConnection("Data Source = flower.db");
        public CSQLite()
        {
            conn.Open();
        }
        public void close()
        {
            conn.Close();
        }

//////////////////////////PLANT/////////////////////////////////////////////////////////////////
////////////////////////////////
        public static List<CPlant> view_plant()
        {
            List<CPlant> cplant = new List<CPlant>(0);
            string query = "SELECT * FROM plant";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CPlant cp = new CPlant();
                cp.id_plant = read[0].ToString();
                cp.name = read[1].ToString();
                cp.categor = read[2].ToString();
                cp.group_ = read[3].ToString();
                cp.country = read[4].ToString();
                cp.price = read[5].ToString();
                cp.counter = read[6].ToString();
                cplant.Add(cp);
            }
            read.Close();
            return cplant;
        }
        public static CPlant view_plant_select(string ID)
        {
            CPlant cplant = new CPlant();
            string query = "SELECT * FROM plant WHERE id_plant = \"" + ID + "\"";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                cplant.id_plant = read[0].ToString();
                cplant.name = read[1].ToString();
                cplant.categor = read[2].ToString();
                cplant.group_ = read[3].ToString();
                cplant.country = read[4].ToString();
                cplant.price = read[5].ToString();
                cplant.counter = read[6].ToString();
            }
            read.Close();
            return cplant;
        }
        public static void insert_plant(CPlant ob, string ID_EMPLOY)
        {
```

```
            string query = "INSERT INTO plant (name, categor, group_, country, price,
counter) " +
                "VALUES (\"" + ob.name + "\", \"" + ob.categor + "\", \"" + ob.group_ + "\",
\"" +
                ob.country + "\", \"" + ob.price + "\", \"" + ob.counter + "\")";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();


            query = "SELECT id_plant FROM plant ORDER BY id_plant DESC LIMIT 1";
            command = new SQLiteCommand(query, conn);
            string n = command.ExecuteScalar().ToString();
            INSERT_HISTORY(ID_EMPLOY, "Додав нові рослини з кодом "+ n, DateTime.Now);
        }
        public static void update_plant(CPlant ob, string ID_EMPLOY)
        {
            string query = "UPDATE plant SET name = \"" + ob.name + "\", categor = \"" +
ob.categor + "\", group_ = \"" + ob.group_ +
                "\", price = \"" + ob.price + "\" WHERE id_plant = \"" + ob.id_plant + "\"";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            INSERT_HISTORY(ID_EMPLOY, "Редагував інформацію про рослини з кодом " +
ob.id_plant, DateTime.Now);
        }
        public static void sum_plant(CPlant ob, string ID_EMPLOY)
        {
            string query = "UPDATE plant SET counter = (SELECT SUM(counter + \"" +
ob.counter + "\") FROM plant WHERE id_plant = \"" + ob.id_plant +
                "\") WHERE id_plant = \"" + ob.id_plant + "\"";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            INSERT_HISTORY(ID_EMPLOY, "Додав " + ob.counter + " рослин з кодом " +
ob.id_plant, DateTime.Now);
        }
        public static bool minus_plant(CPlant ob, string reason, string ID_EMPLOY)
        {
            CPlant cplant = new CPlant();
            cplant = view_plant_select(ob.id_plant);
            if (long.Parse(ob.counter) > long.Parse(cplant.counter)) {
MessageBox.Show("Введена кількість перевищує наявну кількість", "Увага"); return false; }
            else
            {
                string query = "UPDATE plant SET counter = (SELECT SUM(counter - \"" +
ob.counter + "\") FROM plant WHERE id_plant = \"" + ob.id_plant +
                        "\") WHERE id_plant = \"" + ob.id_plant + "\"";
                command = new SQLiteCommand(query, conn);
                command.ExecuteNonQuery();
                CWrite_off cw = new CWrite_off();
                cw.id_plant = ob.id_plant;
                cw.counter = ob.counter;
                cw.date_ = DateTime.Now.ToShortDateString();
                cw.reason = reason;
                insert_write_off(cw, ID_EMPLOY);
                return true;
            }
        }
        public static bool minus_plant(COrder co, string ID_EMPLOY)
        {
            CPlant cplant = new CPlant();
            cplant = view_plant_select(co.id_plant);
            if (long.Parse(co.counter) > long.Parse(cplant.counter)) {
MessageBox.Show("Введена кількість перевищує наявну кількість", "Увага"); return false; }
            else
```

```
            {
                string query = "UPDATE plant SET counter = (SELECT SUM(counter - \"" +
co.counter + "\") FROM plant WHERE id_plant = \"" + co.id_plant +
                        "\") WHERE id_plant = \"" + co.id_plant + "\"";
                command = new SQLiteCommand(query, conn);
                command.ExecuteNonQuery();
                insert_order(co, ID_EMPLOY);
                return true;
            }
        }
        public static List<CPlant> filter_plant(CPlant ob)
        {
            List<CPlant> cplant = new List<CPlant>();
            string query = "SELECT * FROM plant WHERE id_plant LIKE \"%" + ob.id_plant +
"%\" and name LIKE \"%" + ob.name + "%\" and categor LIKE \"%" + ob.categor +
                "%\" and group_ LIKE \"%" + ob.group_ + "%\" and country LIKE \"%" +
ob.country + "%\" and price LIKE \"%" + ob.price +
                "%\" and counter LIKE \"%" + ob.counter + "%\" ORDER BY id_plant";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CPlant cp = new CPlant();
                cp.id_plant = read[0].ToString();
                cp.name = read[1].ToString();
                cp.categor = read[2].ToString();
                cp.group_ = read[3].ToString();
                cp.country = read[4].ToString();
                cp.price = read[5].ToString();
                cp.counter = read[6].ToString();
                cplant.Add(cp);
            }
            read.Close();
            return cplant;
        }

//////////////////////////EMPLOY////////////////////////////////////////////////////////////////
////////////////////////////////
        public static List<CEmploy> view_employ()
        {
            List<CEmploy> cemploy = new List<CEmploy>(0);
            string query = "SELECT * FROM employ";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CEmploy ce = new CEmploy();
                ce.id_employ = read[0].ToString();
                ce.password = read[1].ToString();
                ce.full_name = read[2].ToString();
                ce.position = read[3].ToString();
                ce.birthday = read[4].ToString();
                ce.address = read[5].ToString();
                ce.phone = read[6].ToString();
                ce.mail = read[7].ToString();
                cemploy.Add(ce);
            }
            read.Close();
            return cemploy;
        }
        public static CEmploy view_employ_select(string ID)
        {
            CEmploy ce = new CEmploy();
            string query = "SELECT * FROM employ WHERE id_employ = \"" + ID + "\"";
            command = new SQLiteCommand(query, conn);
```

```
            read = command.ExecuteReader();
            while (read.Read())
            {
                ce.id_employ = read[0].ToString();
                ce.password = read[1].ToString();
                ce.full_name = read[2].ToString();
                ce.position = read[3].ToString();
                ce.birthday = read[4].ToString();
                ce.address = read[5].ToString();
                ce.phone = read[6].ToString();
                ce.mail = read[7].ToString();
            }
            read.Close();
            return ce;
        }
        public static bool password_employ(string employ, string password)
        {
            bool flag = false;
            string query = "SELECT id_employ FROM employ WHERE id_employ = \"" + employ +
"\"and password = \"" + password + "\"";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                if (read[0].ToString() == employ) flag = true;
            }
            read.Close();
            return flag;
        }
        public static void open(string employ)
        {
            INSERT_HISTORY(employ, "Робітник з кодом " + employ + " увійшов до системи",
DateTime.Now);
        }
        public static void exit(string employ)
        {
            INSERT_HISTORY(employ, "Робітник з кодом " + employ + " завершив сеанс",
DateTime.Now);
        }
        public static void update_employ_password(string employ, string password, string
ID_EMPLOY)
        {
            string query = "UPDATE employ SET password = \"" + password + "\" WHERE
id_employ = \"" + employ + "\"";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            INSERT_HISTORY(ID_EMPLOY, "Робітник змінив свій пароль", DateTime.Now);
        }
        public static bool insert_employ(CEmploy ob, string ID_EMPLOY)
        {
            if(select_mail_phone(null, null, ob.phone, ob.mail) == 1)
            {
                MessageBox.Show("Робітник с таким номером телефону вже існує","Увага");
                return false;
            }
            if (select_mail_phone(null, null, ob.phone, ob.mail) == 2)
            {
                MessageBox.Show("Робітник з такою електронною поштою вже існує", "Увага");
                return false;
            }
            string query = "INSERT INTO employ (password, full_name, position, birthday,
address, phone, mail) " +
                "VALUES (\"" + ob.password + "\", \"" + ob.full_name + "\", \"" +
ob.position + "\", \"" +
```

```
                ob.birthday + "\", \"" + ob.address + "\", \"" + ob.phone + "\", \"" +
ob.mail + "\")";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            query = "SELECT id_employ FROM employ ORDER BY id_employ DESC LIMIT 1";
            command = new SQLiteCommand(query, conn);
            string n = command.ExecuteScalar().ToString();
            INSERT_HISTORY(ID_EMPLOY, "Додав нового робітника з кодом "+ n, DateTime.Now);
            return true;
        }
        public static bool update_employ(string _phone, string _mail, CEmploy ob, string
ID_EMPLOY)
        {
            if (select_mail_phone(_phone, _mail, ob.phone, ob.mail) == 1)
            {
                MessageBox.Show("Робітник с таким номером телефону вже існує", "Увага");
                return false;
            }
            if (select_mail_phone(_phone, _mail, ob.phone, ob.mail) == 2)
            {
                MessageBox.Show("Робітник з такою електронною поштою вже існує", "Увага");
                return false;
            }
            string query = "UPDATE employ SET password = \"" + ob.password + "\", full_name
= \"" + ob.full_name + "\", position = \"" + ob.position +
                "\", birthday = \"" + ob.birthday +
                "\", address = \"" + ob.address + "\", phone = \"" + ob.phone + "\", mail =
\"" + ob.mail + "\" WHERE id_employ = \"" + ob.id_employ + "\"";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            INSERT_HISTORY(ID_EMPLOY, "Редагована інформацію робітника з кодом "+
ob.id_employ, DateTime.Now);
            return true;
        }
        public static void delete_employ(string id_employ, string ID_EMPLOY)
        {
            if(id_employ == ID_EMPLOY) { MessageBox.Show("Ви не может видалити самого себе",
"Увага"); return; }
            string query = "DELETE FROM employ WHERE id_employ = \"" + id_employ + "\"";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();
            INSERT_HISTORY(ID_EMPLOY, "Видалено робітника з  кодом " + id_employ,
DateTime.Now);
        }
        public static List<CEmploy> filter_employ(CEmploy ob)
        {
            List<CEmploy> cemploy = new List<CEmploy>();
            string query = "SELECT * FROM employ WHERE id_employ LIKE \"%" + ob.id_employ +
"%\" and full_name LIKE \"%" + ob.full_name +
                "%\" and position LIKE \"%" + ob.position + "%\" and birthday LIKE \"%" +
ob.birthday + "%\" and address LIKE \"%" + ob.address +
                "%\" and phone LIKE \"%" + ob.phone + "%\" and mail LIKE \"%" + ob.mail +
"%\" ORDER BY id_employ";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CEmploy ce = new CEmploy();
                ce.id_employ = read[0].ToString();
                ce.password = read[1].ToString();
                ce.full_name = read[2].ToString();
                ce.position = read[3].ToString();
                ce.birthday = read[4].ToString();
```

```
                ce.address = read[5].ToString();
                ce.phone = read[6].ToString();
                ce.mail = read[7].ToString();
                cemploy.Add(ce);
            }
            read.Close();
            return cemploy;
        }
        public static List<CEmploy> select_employ_meneger()
        {
            List<CEmploy> cemploy = new List<CEmploy>();
            string query = "SELECT * FROM employ WHERE position = \"Менеджер\"";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CEmploy ce = new CEmploy();
                ce.id_employ = read[0].ToString();
                ce.password = read[1].ToString();
                ce.full_name = read[2].ToString();
                ce.position = read[3].ToString();
                ce.birthday = read[4].ToString();
                ce.address = read[5].ToString();
                ce.phone = read[6].ToString();
                ce.mail = read[7].ToString();
                cemploy.Add(ce);
            }
            read.Close();
            return cemploy;
        }
        public static int select_mail_phone(string _phone, string _mail, string phone,
string mail)
        {
            List<CEmploy> cemploy = new List<CEmploy>(0);
            string query = "SELECT phone, mail FROM employ";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CEmploy ce = new CEmploy();
                ce.phone = read[0].ToString();
                ce.mail = read[1].ToString();
                cemploy.Add(ce);
            }
            read.Close();
            for (int i = 0; i < cemploy.Count; i++) if (cemploy[i].phone == phone && !(phone
== _phone)) return 1;
                else if (cemploy[i].mail == mail && !(mail == _mail)) return 2;
            return 0;
        }

//////////////////////ORDER/////////////////////////////////////////////////////////////
////////////////////////////////
        public static COrder view_order_select()
        {
            COrder co = new COrder();
            string query = "SELECT * FROM order_ ORDER BY id_order DESC LIMIT 1";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                co.id_order = read[0].ToString();
                co.id_plant = read[1].ToString();
                co.id_employ = read[2].ToString();
                co.date_ = read[3].ToString();
```

```
                co.price = read[4].ToString();
                co.counter = read[5].ToString();
                co.suma = read[6].ToString();
            }
            read.Close();
            return co;
        }
        public static List<COrder> view_order()
        {
            List<COrder> corder = new List<COrder>(0);
            string query = "SELECT * FROM order_";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                COrder co = new COrder();
                co.id_order = read[0].ToString();
                co.id_plant = read[1].ToString();
                co.id_employ = read[2].ToString();
                co.date_ = read[3].ToString();
                co.price = read[4].ToString();
                co.counter = read[5].ToString();
                co.suma = read[6].ToString();
                corder.Add(co);
            }
            read.Close();
            return corder;
        }
        public static void insert_order(COrder ob, string ID_EMPLOY)
        {
            string query = "INSERT INTO order_ (id_plant, id_employ, date_, price, counter,
suma) " +
                "VALUES (\"" + ob.id_plant + "\", \"" + ob.id_employ + "\", \"" + ob.date_ +
"\", \"" + ob.price + "\", \"" + ob.counter + "\", \"" + ob.suma + "\")";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            query = "SELECT id_order FROM order_ ORDER BY id_order DESC LIMIT 1";
            command = new SQLiteCommand(query, conn);
            string n = command.ExecuteScalar().ToString();
            INSERT_HISTORY(ID_EMPLOY, "Оформив замовлення з кодом " + n, DateTime.Now);
        }
        public static List<COrder> filter_order(COrder ob)
        {
            List<COrder> orders = new List<COrder>();
            string query = "SELECT * FROM order_ WHERE id_order LIKE \"%" + ob.id_order +
"%\" and id_plant LIKE \"%" + ob.id_plant +
                "%\" and id_employ LIKE \"%" + ob.id_employ + "%\" and date_ LIKE \"%" +
ob.date_ + "%\" and price LIKE \"%" + ob.price +
                "%\" and counter LIKE \"%" + ob.counter + "%\" and suma LIKE \"%" + ob.suma
+ "%\" ORDER BY id_order";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                COrder co = new COrder();
                co.id_order = read[0].ToString();
                co.id_plant = read[1].ToString();
                co.id_employ = read[2].ToString();
                co.date_ = read[3].ToString();
                co.price = read[4].ToString();
                co.counter = read[5].ToString();
                co.suma = read[6].ToString();
                orders.Add(co);
            }
```

```
            read.Close();
            return orders;
        }

//////////////////////HISORY//////////////////////////////////////////////////////////////
////////////////////////////////
        public static List<CHistory> view_history()
        {
            List<CHistory> chistory = new List<CHistory>(0);
            string query = "SELECT history.*, employ.full_name FROM history, employ WHERE
history.id_employ = employ.id_employ";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CHistory ch = new CHistory();
                ch.id_record = read[0].ToString();
                ch.id_employ = read[1].ToString();
                ch.date_time = read[2].ToString();
                ch.record_history = read[3].ToString();
                ch.name = read[4].ToString();
                chistory.Add(ch);
            }
            read.Close();
            return chistory;
        }
        public static List<CHistory> view_history_admin(string ID_EMPLOY)
        {
            List<CHistory> chistory = new List<CHistory>(0);
            string query = "SELECT history.*, employ.full_name FROM history, employ WHERE
history.id_employ = employ.id_employ and history.id_employ = \"" + ID_EMPLOY + "\"";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CHistory ch = new CHistory();
                ch.id_record = read[0].ToString();
                ch.id_employ = read[1].ToString();
                ch.date_time = read[2].ToString();
                ch.record_history = read[3].ToString();
                ch.name = read[4].ToString();
                chistory.Add(ch);
            }
            read.Close();
            return chistory;
        }
        public static void INSERT_HISTORY(string id_employ, string record_history, DateTime
dateTime)
        {
            string query = "INSERT INTO history (id_employ, date_time, record_history) " +
                "VALUES (\"" + id_employ + "\", \"" + dateTime + "\", \"" + record_history +
"\")";
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();
        }
        public static List<CHistory> filter_history(CHistory ob)
        {
            List<CHistory> histories = new List<CHistory>();
            string query = "SELECT history.*, employ.full_name FROM history, employ WHERE
history.id_employ = employ.id_employ and " +
                "history.id_record LIKE \"%" + ob.id_record + "%\" and history.id_employ
LIKE \"%" + ob.id_employ +
                "%\" and employ.full_name LIKE \"%" + ob.name + "%\"and history.date_time
LIKE \"%" + ob.date_time +
```

```
                "%\" and history.record_history LIKE \"%" + ob.record_history + "%\" ORDER
BY history.id_record";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CHistory ch = new CHistory();
                ch.id_record = read[0].ToString();
                ch.id_employ = read[1].ToString();
                ch.date_time = read[2].ToString();
                ch.record_history = read[3].ToString();
                ch.name = read[4].ToString();
                histories.Add(ch);
            }
            read.Close();
            return histories;
        }

//////////////////////////WRITE_OFF/////////////////////////////////////////////////////////////
////////////////////////////////////
        public static List<CWrite_off> view_write_off()
        {
            List<CWrite_off> cwrite_off = new List<CWrite_off>(0);
            string query = "SELECT write_off.id_write_off, write_off.id_plant, plant.name,
plant.categor, plant.group_, plant.country," +
                "write_off.counter, write_off.date_, write_off.reason FROM plant, write_off
WHERE write_off.id_plant = plant.id_plant";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CWrite_off cw = new CWrite_off();
                cw.id_write_off = read[0].ToString();
                cw.id_plant = read[1].ToString();
                cw.name = read[2].ToString();
                cw.categor = read[3].ToString();
                cw.group_ = read[4].ToString();
                cw.country = read[5].ToString();
                cw.counter = read[6].ToString();
                cw.date_ = read[7].ToString();
                cw.reason = read[8].ToString();
                cwrite_off.Add(cw);
            }
            read.Close();
            return cwrite_off;
        }
        public static void insert_write_off(CWrite_off ob, string ID_EMPLOY)
        {
            string query = string.Format("INSERT INTO write_off (id_plant, counter, date_,
reason) " +
                "VALUES (\"" + ob.id_plant + "\", \"" + ob.counter + "\", \"" + ob.date_ +
"\", \"{0}\")", ob.reason);
            command = new SQLiteCommand(query, conn);
            command.ExecuteNonQuery();

            query = "SELECT id_write_off FROM write_off ORDER BY id_write_off DESC LIMIT 1";
            command = new SQLiteCommand(query, conn);
            string n = command.ExecuteScalar().ToString();
            INSERT_HISTORY(ID_EMPLOY, "Списав рослини, код списання " + n, DateTime.Now);
        }
        public static List<CWrite_off> filter_write_off(CWrite_off ob)
        {
            List<CWrite_off> cwrite_off = new List<CWrite_off>();
            string query = "SELECT write_off.id_write_off, write_off.id_plant, plant.name,
plant.categor, plant.group_, plant.country," +
```

```
                "write_off.counter, write_off.date_, write_off.reason FROM plant, write_off
WHERE write_off.id_plant = plant.id_plant and " +
                "write_off.id_plant LIKE \"%" + ob.id_plant + "%\" and plant.name LIKE \"%"
+ ob.name + "%\" and plant.categor LIKE \"%" + ob.categor +
                "%\" and plant.group_ LIKE \"%" + ob.group_ + "%\" and plant.country LIKE
\"%" + ob.country + "%\" and write_off.counter LIKE \"%" + ob.counter +
                "%\" and write_off.date_ LIKE \"%" + ob.date_ + "%\" and write_off.reason
LIKE \"%" + ob.reason + "%\" ORDER BY write_off.id_write_off";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CWrite_off cw = new CWrite_off();
                cw.id_write_off = read[0].ToString();
                cw.id_plant = read[1].ToString();
                cw.name = read[2].ToString();
                cw.categor = read[3].ToString();
                cw.group_ = read[4].ToString();
                cw.country = read[5].ToString();
                cw.counter = read[6].ToString();
                cw.date_ = read[7].ToString();
                cw.reason = read[8].ToString();
                cwrite_off.Add(cw);
            }
            read.Close();
            return cwrite_off;
        }

//////////////////////////PROGNOZ/////////////////////////////////////////////////////////////
///////////////////////////////////
        public static List<string> prognoz_count_plant()
        {
            List<string> mas = new List<string>();
            string query = "SELECT date_ FROM order_";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read()) mas.Add(read[0].ToString());
            read.Close();
            return mas;
        }
        public static List<string> prognoz_count_plant_is_employ(string id)
        {
            List<string> mas = new List<string>();
            string query = "SELECT date_ FROM order_ WHERE id_employ = \"" + id + "\"";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read()) mas.Add(read[0].ToString());
            read.Close();
            return mas;
        }
        public static List<CEmploy> filter_prognoz(string id, string name)
        {
            List<CEmploy>arr = new List<CEmploy>(0);
            string query = "SELECT id_employ, full_name FROM employ WHERE position =
\"Менеджер\" and " +
                "id_employ LIKE \"%" + id + "%\" and full_name LIKE \"%" + name + "%\" ORDER
BY id_employ";
            command = new SQLiteCommand(query, conn);
            read = command.ExecuteReader();
            while (read.Read())
            {
                CEmploy ce = new CEmploy();
                ce.id_employ = (read[0].ToString());
                ce.full_name = (read[1].ToString());
                arr.Add(ce);
```

```
            }
            read.Close();
            return arr;
        }

//////////////////////////ADMIN/////////////////////////////////////////////////////////////////
////////////////////////////////
        public static string GetAdmin(string ID)
        {
            string query = "SELECT position FROM employ WHERE id_employ = \"" + ID + "\"
ORDER BY id_employ";
            command = new SQLiteCommand(query, conn);
            return command.ExecuteScalar().ToString();
        }
    }
}
```

InputMessage.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace InputBox
{
    public class InputBox : Form
    {
        private readonly TextBox _textBox;

        public InputBox(string title = "", string labeltext = "", bool isDigits = false)
        {
            StartPosition = FormStartPosition.CenterScreen;
            MaximizeBox = false;
            MinimizeBox = false;
            FormBorderStyle = FormBorderStyle.Fixed3D;
            Size = new Size(300, 150);
            Text = title;

            _textBox = new TextBox
            {
                Size = new Size(250, 25),
                Font = new Font(DefaultFont, FontStyle.Regular),
                Location = new Point(20, 50),
                Text = "",
                MaxLength = 10
            };

            if (isDigits)
            {
                _textBox.KeyPress += SetOnlyDigits;
            }

            if (isDigits == false)
            {
                _textBox.KeyPress += SetOnlyStr;
            }

            Controls.Add(_textBox);

            _textBox.MaxLength = 10;
            _textBox.Show();

            _textBox.KeyPress += textBox_KeyPress;

            var label = new Label
```

```
    {
        AutoSize = false,
        Size = new Size(250, 25)
    };
    label.Font = new Font(label.Font, FontStyle.Regular);
    label.Location = new Point(20, 25);
    label.Text = labeltext;

    Controls.Add(label);

    label.Show();

    var buttonOk = new Button
    {
        Size = new Size(80, 25),
        Location = new Point(105, 75),
        DialogResult = DialogResult.OK,
        Text = "OK"
    };

    Controls.Add(buttonOk);

    buttonOk.Show();

    var buttonCancel = new Button
    {
        Size = new Size(80, 25),
        Location = new Point(190, 75),
        Text = "Вийти"
    };

    Controls.Add(buttonCancel);

    buttonCancel.Show();

    buttonCancel.Click += buttonCancel_Click;
}

public void textBox_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar != (char)Keys.Enter)
    {
        return;
    }
    DialogResult = DialogResult.OK;

    Close();
}

public void buttonCancel_Click(object sender, EventArgs e)
{
    Close();
}

public string GetString()
{
    return ShowDialog() != DialogResult.OK ? null : _textBox.Text;
}

public void SetOnlyDigits(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar <= 47 || e.KeyChar >= 59) && e.KeyChar != 8)
        e.Handled = true;
}
public void SetOnlyStr(object sender, KeyPressEventArgs e)
```

```
        {
            if ((e.KeyChar >= 33 && e.KeyChar <= 38) || (e.KeyChar >= 40 && e.KeyChar <= 44)
|| (e.KeyChar >= 46 && e.KeyChar <= 64) ||
                (e.KeyChar >= 91 && e.KeyChar <= 96) || (e.KeyChar >= 123 && e.KeyChar <=
127)) e.Handled = true;
        }

        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // InputBox
            //
            this.ClientSize = new System.Drawing.Size(284, 261);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            this.Name = "InputBox";
            this.ResumeLayout(false);

        }
    }
}
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_ Янишівський.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_ Янишівський.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Програма_Янишівський.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Презентація_Янишівський.ppt	Презентація кваліфікаційної роботи