КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Інформаційна система керування запасами на складі»

Виконав_____
(Підпис)
Чуприна Павло Степанович
(прізвище, ім'я, по батькові)

Керівник Гарко Ірина Ігорівна
(прізвище, ім'я, по батькові)

(Резолюція «До захисту»

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри ___ ____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Унікальність тексту - %

Київ – 2022

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	21.12.2022	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2022	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	30.03.2022	
9.	Подання роботи у першому варіанті	28.04.2022	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	23.05.2022	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	10.06.2022	
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022 24.06.2022	

Здобувач вищої освіти

Керівник

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Складові частини кваліфікаційної роботи бакалавра	Обсяг, арк.
Титульний аркуш	1
Календарний план кваліфікаційної роботи бакалавра	1
Відомість кваліфікаційної роботи бакалавра	1
Анотація	1
Анотація (іноземною мовою - англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	12
2	6
3	12
Висновки	2
Список використаних джерел	3
Додатки	9

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Чуприна П.С.			Відомість кваліфікаційної роботи бакалавра	Лист	Листів
Керівн.	Гарко І.І.					
Н/контр.	Базилюк А.М.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Кваліфікаційна робота бакалавра містить: 51 сторінка, 28 рисунків, 28 використаних джерел, 1 додаток. Робота присвячена створенню застосунку для полегшення керування запасами на складі із використанням мови C# за допомогою інтерфейсу програмування застосунків *Windows Form.* Метою кваліфікаційної роботи бакалавра є підвищення практичності та ефективності керування запасами на складі. Об'єкт дослідження є процеси керування запасами на складі. Предметом дослідження є програмні засади розроблення програмної системи керування запасами на складі. Для досягнення поставленої мети треба вирішити такі завдання: дослідити переваги та недоліки систем-конкурентів; здійснити аналіз програмно-технологічних рішень побудови схожих застосунків; створити простий та доступний застосунок для керування запасами на складі з урахуванням проведених досліджень; розробити покрокову інструкцію користувача для здійснення базових операцій. Методи дослідження: теорія управління систем для вивчення сучасних методів побудови систем контролю запасів товарів, аналогія залучені в процесі проектування, розроблення та побудови власної системи керування запасами на складі, метод порівняння, що застосовано для аналізу наявних ресурсів та програмних систем контролю товару.

Ключові слова: програмна система, застосунок, система керування запасами на складі.

ABSTRACT

The bachelor's thesis contains: 51 pages, 28 drawings, 28 sources used, and 1 appendix. This thesis is devoted to the creation of applications for making easier the control of inventory in stock using the C# language and the programming interface of Windows Form applications. The aim of the work is building a software system of inventory control in the warehouse. The object of research of the bachelor's thesis is the ways to facilitate product control in warehouses. The subject of research is software, organizational principles of development of software inventory management system in the warehouse. To achieve this goal you need to solve the following tasks: explore the advantages and disadvantages of competing systems; analyze software and technological solutions for building similar applications; design, implement an application for inventory control in the warehouse, taking into account the research. Research methods: systems management theory used in the study of examples of modern methods of building product control systems, the analogy involved in the design, development and construction of its own product control system, the method of comparison used to analyze available resources and software product control systems.

Keywords: software system, application, product management system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

VS – Visual Studio

WPF – Windows Presentation Foundation

AP – Adobe Photoshop

BDS – Base Data Access

XAML – Extensible Application Markup Language

VGA – Video Graphics Array

WF – Windows Form

IDE – Integrated Development Environment

OS – Operating System

OOP – Object-oriented Programming

PS – Programming Support

ВСТУП

Актуальність теми. На сьогоднішній день, завдяки стрибку у розвитку комп'ютерної техніки та програмного забезпечення відбулося розширення сфер застосування персональних комп'ютерів. Програмні системи сьогодні присутні всюди: практично будь-які електронні пристрої містять програмне забезпечення того чи іншого виду. Без відповідного програмного забезпечення в сучасному світі неможливо уявити індустріальне виробництво, системи оборони держави, охорони здоров'я, фінансові й урядові заклади. У різноманітних системах вартість програмного забезпечення складає більшу частини вартості виробу технічної системи. А провідною галуззю в суспільстві є галузь розробки комп'ютерних програм, що є важливою складовою науково-технологічного процесу.

Мета роботи: підвищення практичності та ефективності керування запасами на складі. Виходячи з мети роботи, виникають наступні **завдання:**

- дослідити переваги та недоліки систем-конкурентів;
- здійснити аналіз програмно-технологічних рішень побудови схожих застосунків;
- створити простий та доступний застосунок для керування запасами на складі з урахуванням проведених досліджень;
- розробити покрокову інструкцію користувача для здійснення базових операцій.

Об'єктом дослідження є процеси керування запасами на складі.

Предметом дослідження є програмні засади розроблення програмної системи керування запасами на складі.

Методи дослідження. Теорія управління систем для вивчення сучасних методів побудови систем контролю запасів товарів, аналогія залучені в процесі проектування, розроблення та побудови власної системи керування запасами на

складі, метод порівняння, що застосовано для аналізу наявних ресурсів та програмних систем контролю товару.

Практичне значення одержаних результатів полягає в створенні робочого застосунку керування запасами на складі, з можливістю майбутнього його удосконалення.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків до розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 51 сторінка. Список використаних джерел охоплює 28 найменувань.

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО РОЗРОБЛЕННЯ І ВПРОВАДЖЕННЯ СИСТЕМ КЕРУВАННЯ ЗАПАСАМИ ПРОДУКЦІЇ

1.1 Теоретичні основи проектування та реалізації програмних продуктів

Програмна інженерія – це наука побудови комп'ютерних програмних систем на інженерній основі за методами, засобами і інструментами програмування, сучасними стандартами процесів життєвих циклів, менеджменту та керування якістю [1]. Особливістю виробництва нових систем є технологія їх проектування від аналізу предметної області до утворення коду для виконання на комп'ютерах. Основа інженерії проектування – теорія алгоритмів і програмування, теорія обчислень і розподіленої обробки, теорія обчислювальних мереж та ін.

Проектування у програмній інженерії – це конструювання комп'ютерних систем методами та засобами програмування за такими загальними кроками [2]:

- опис вимог;
- опис специфікацій системи;
- розроблення системи;
- тестування, оцінка надійності і якості системи.

Рисунок 1.1 – Головні області програмної інженерії

На рис. 1.1 зображено головні області програмної інженерії [3], кожна з яких відіграє важливу роль в розробленні програмного забезпечення.

1.2 Аналіз предметної області та існуючих підходів до вирішення завдання

Одним з головних етапів проектування будь-якого типу програмних застосунків є аналіз предметної області, що закінчується побудовою інформаційної системи. В інтернеті є програми, які дозволяють підприємцям вести облік в роздрібній торгівлі, але більшість з них є «сирими» або малофункціональними. Після складання списку відповідних програм перед підприємцем постає складне питання: за якими критеріями їх оцінювати?

Нижче перераховані критерії вибору програм обліку, які потрібно врахувати підприємцю:

- Перелік операцій,які підтримуються. Комусь потрібно знати просто прихід / витрата, а для когось важливим при прийнятті остаточного рішення є додатковий ціновий облік і аналітика продажів.
- Вартість впровадження та супроводу. Ціна повинна бути збалансованою і доступною як для великих компаній, так і для невеликих підприємств. Найочевиднішим рішенням для врівноваження цінової політики є впровадження щомісячних абонплат, які передбачатимуть різну кількість функціоналу залежно від заплачених коштів.
- Цілодобова технічна підтримка.
- Наявність додаткових опціональних модулів (CMS, бухгалтерських, логістичних). При плануванні дисконтної системи CMS просто необхідна.
- Мережеві можливості.
- Простота освоєння. Цей критерій є одним з найважливіших при розробленні застосунку. Інтерфейс не повинен бути перенавантаженим. Новий співробітник повинен за кілька годин освоїти основні можливості програми.

- Стабільність роботи. Програма не повинна підвисати і перезавантажуватися, адже це може впливати на стабільність прийому та видачі на складах.
- Наявність повнофункціональної демо-версії. Набагато простіше вибрати програму, завантаживши її повну версію і випробувавши можливості.
- Відкритий API, що дозволяє допрацьовувати програму під індивідуальні потреби клієнта.
- Зручний інтерфейс. Перемикання між меню під час роботи повинно займати у персоналу мінімум часу.

Проводячи аналіз існуючих підходів, було виявлено наступні застосунки: МойСклад, LiteBox, SUBTOTAL, Qasl, «1С: Торговля и склад». Розглянемо детальніше кожен із них.

МойСклад – комерційний SaaS продукт для управління торгівлею і складського обліку, призначений для автоматизації малого і середнього бізнесу. Сервіс реалізує функції обробки замовлень, управління продажами і закупівлями, складського обліку та контролю фінансових розрахунків. МойСклад став одним з перших сервісів SaaS, розроблених в Росії і призначених для російського ринку.

Переваги програми «МойСклад»:

- широкий функціонал, який підходить для роздрібної, оптової торговлі, сфери громадського харчування и невеликих виробництв;
- підтримка дисконтних карт, створення клієнтської бази, формування воронки продаж;
- стабільна робота;
- дружелюбний інтерфейс і легкість освоєння програми новими співробітниками;
- є демо-версії з повним функціоналом;
- кросплатформність: програмою можна користуватися на ОС Windows, macOS, Android, Linux, iOS;
- відкритий API.

Недоліки програми «МойСклад»:

- вартість щомісячної абонплати вище середньостатистичної;
- відсутність шаблонів для продаж.

На рис. 1.2 зображено основний інтерфейс програми “Мой Склад”.

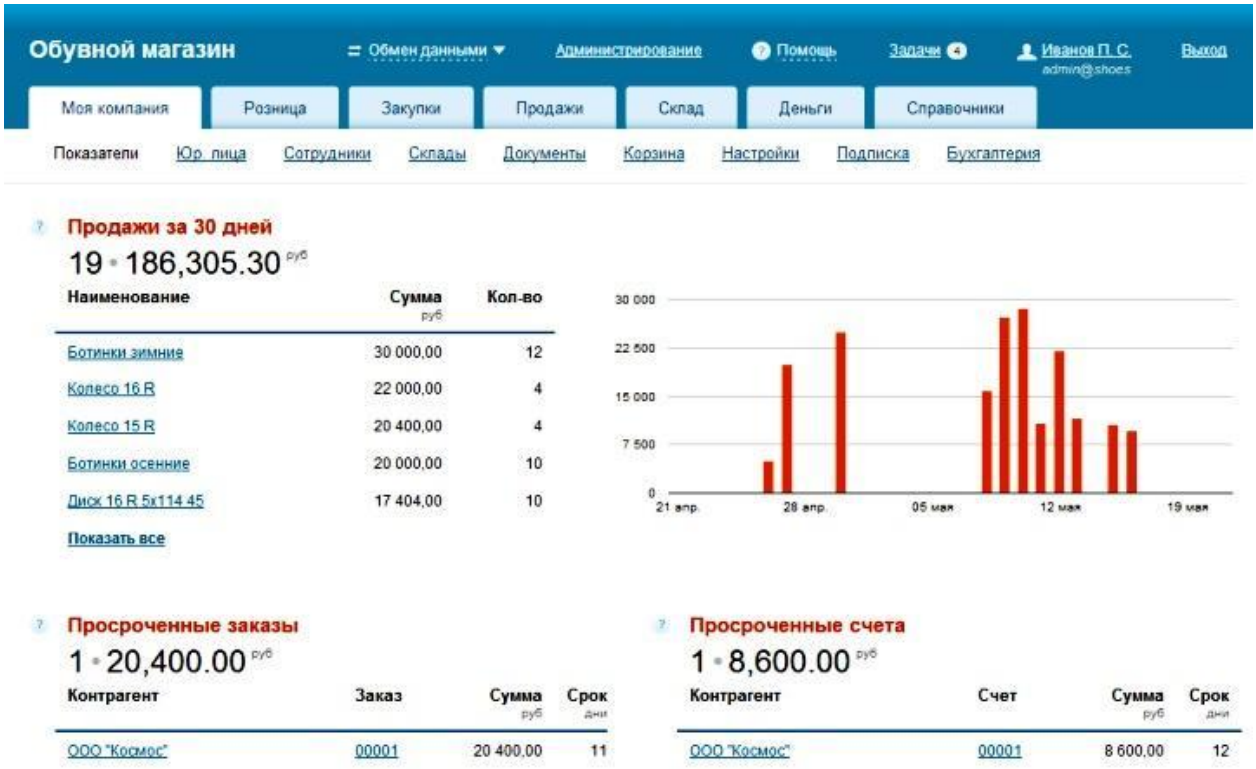

Рисунок 1.2 – Інтерфейс «МойСклад»

На рис. 1.3 зображено розділ “Продажи” програми “Мой Склад”.

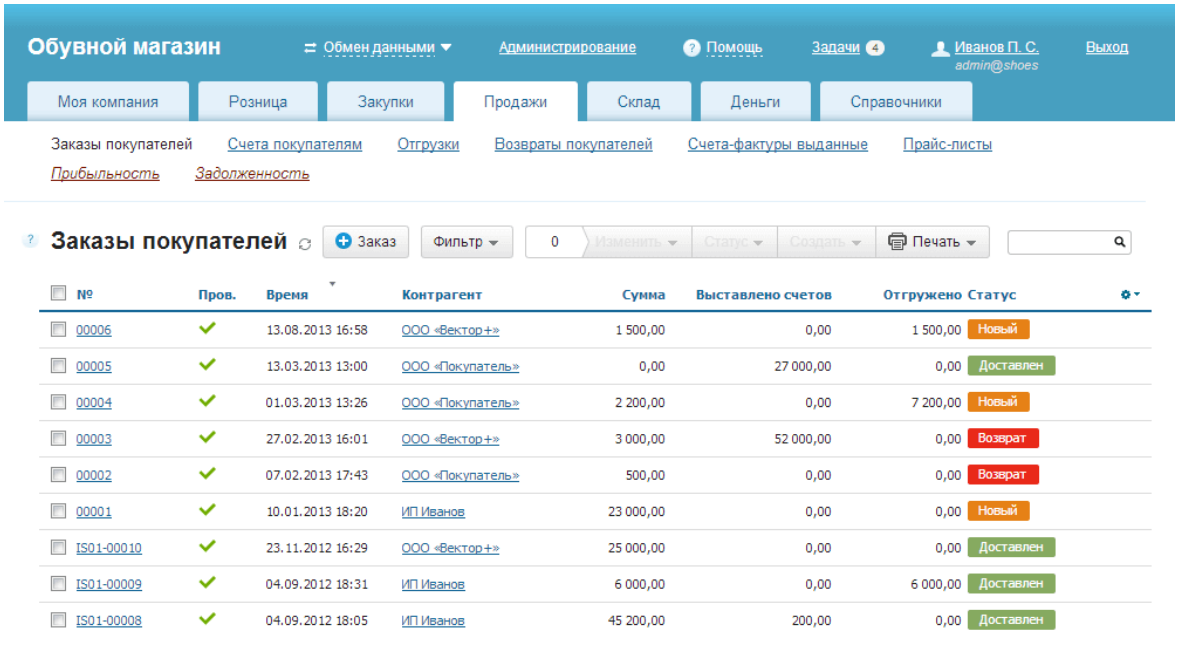

Рисунок 1.3 – Розділ «Продажи» програми «Мой Склад»

На рис. 1.4 зображено Статистику програми “Мой Склад”.

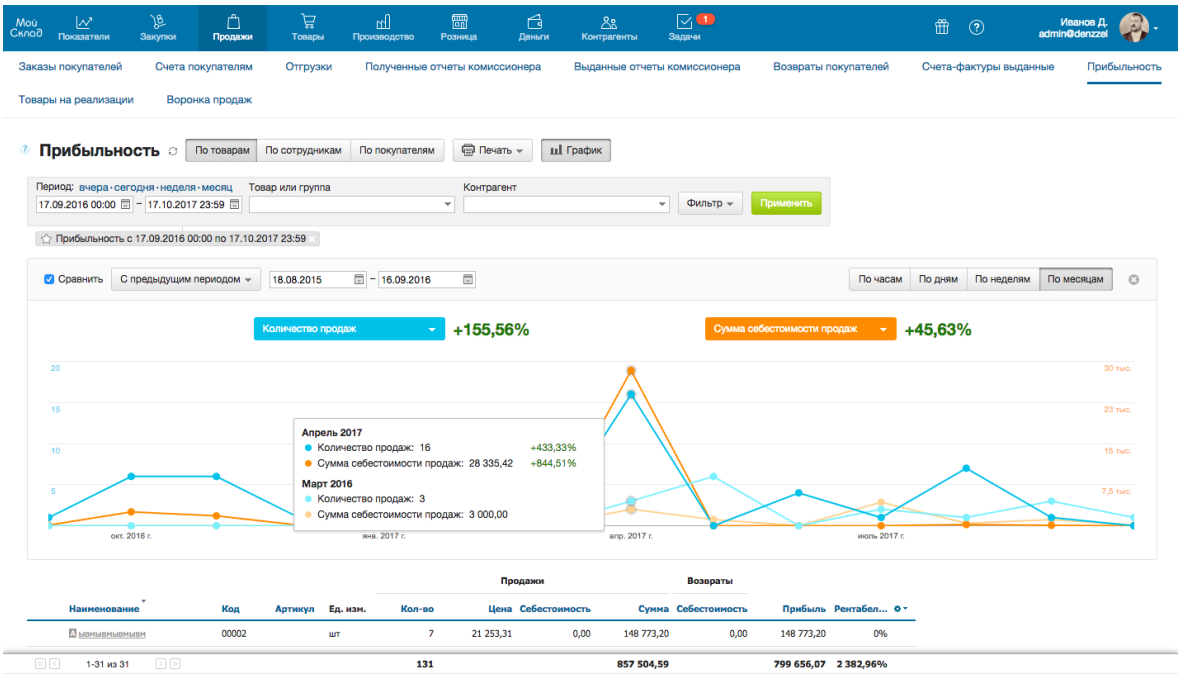

Рисунок 1.4 – Статистика «Мой Склад»

LiteBox – повноцінна програма для автоматизації магазину роздрібної торгівлі з широким функціоналом можливостей, завдяки якій ви максимально швидко і зручно зможете організувати роботу свого магазину, використовуючи якісний та інтуїтивно зрозумілий інтерфейс управління.

Функціонал програми представлений шістьма напрямками:

- управління торгівлею;
- складський облік;
- аналітичні звіти;
- управління закупівлями;
- документи;
- маркетингові інструменти.

На рис. 1.5 зображено основний інтерфейс програми “LiteBox”.

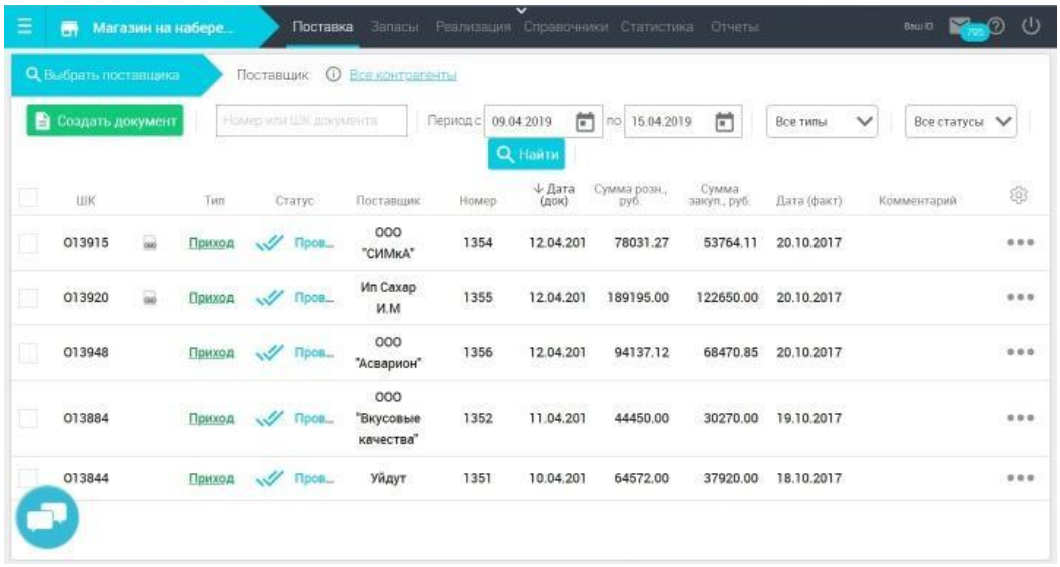

Рисунок 1.5 – Інтерфейс «LiteBox»

На рис. 1.6 зображено статистику у програмі “LiteBox”.

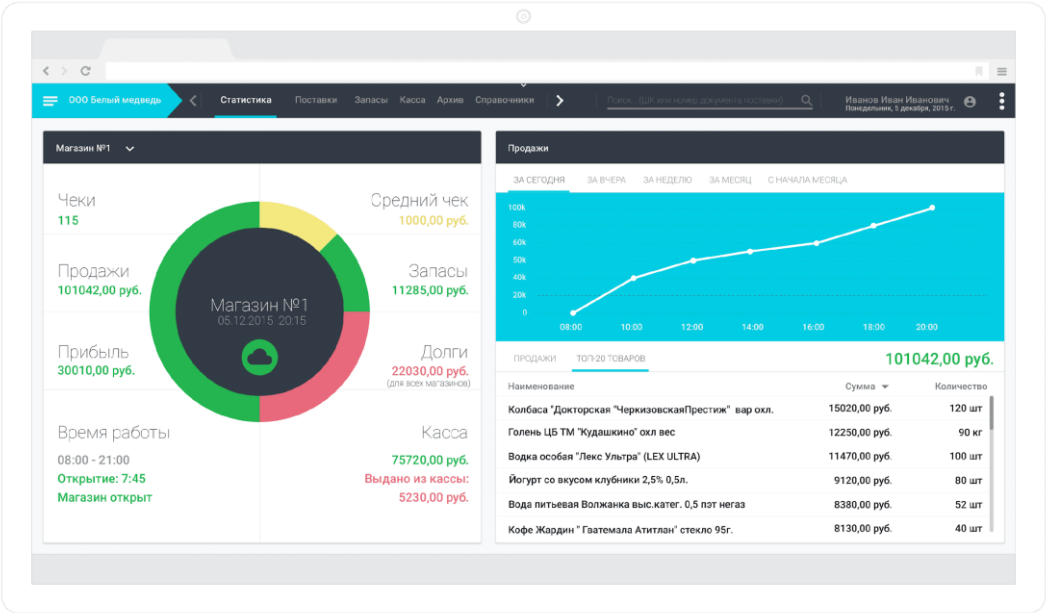

Рисунок 1.6 – Статистика «LiteBox»

Subtotal – програма автоматизації магазину роздрібної торгівлі, підходить для громадського харчування, сфери послуг і продажу. Легка інтеграція з касою.

На рис. 1.7 зображено інтерфейс програми “SUBTOTAL”.

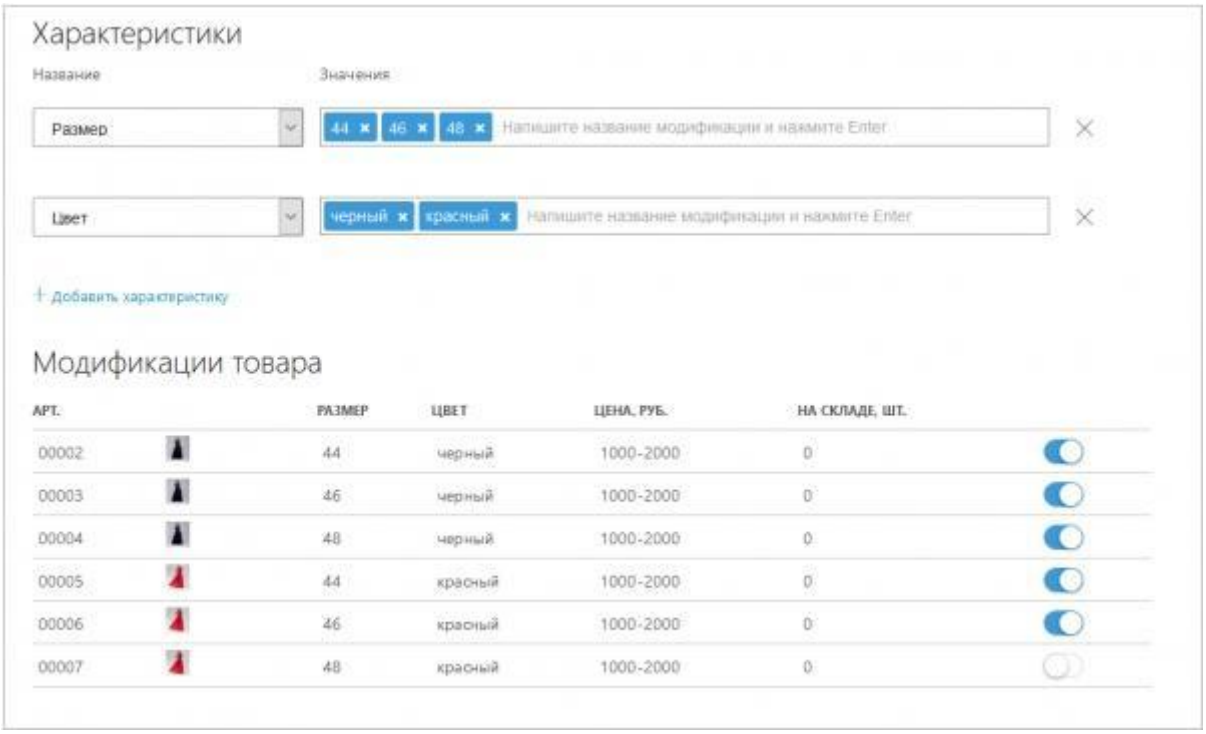

Рисунок 1.7 – Інтерфейс «SUBTOTAL»

Переваги SUBTOTAL:

- автоматична передача даних ,що стосуються продажу в 1С, і навпаки, перенесення відомостей про товари з 1С в хмарну базу Sigma;
- наявність детальних відео і фото інструкцій по роботі з кожним меню програми;
- річні тарифи на обслуговування нижчі за середньоринкові;
- складання технологічних карт для громадського харчування;
- простий, інтуїтивно зрозумілий інтерфейс, що дозволяє швидко навчати роботі з програмою нових співробітників;
- повноцінний модуль бази клієнтів з можливістю прив’язки дисконтних карт;
- наявність демонстраційної версії з безкоштовним двотижневим періодом;
- підключення декількох магазинів і складів.

Недоліки SUBTOTAL:

- недостатній функціонал на найнижчому тарифі;
- відсутність підтримки продажів через інтернет-магазини;
- робота на комп'ютері тільки через браузер;
- програма максимально розкриває свої можливості тільки з онлайн-касою виробника АТОЛ, хоча підтримує пристрої та інших компаній;
- АТОЛ Sigma 10 працює тільки з ПО Sigma;
- відсутні функції резервування товарів, управління доставкою;
- неможливість email-розсилок;
- відсутність телефону підтримки.

На рис. 1.8 зображена статистика у програмі "SUBTOTAL".

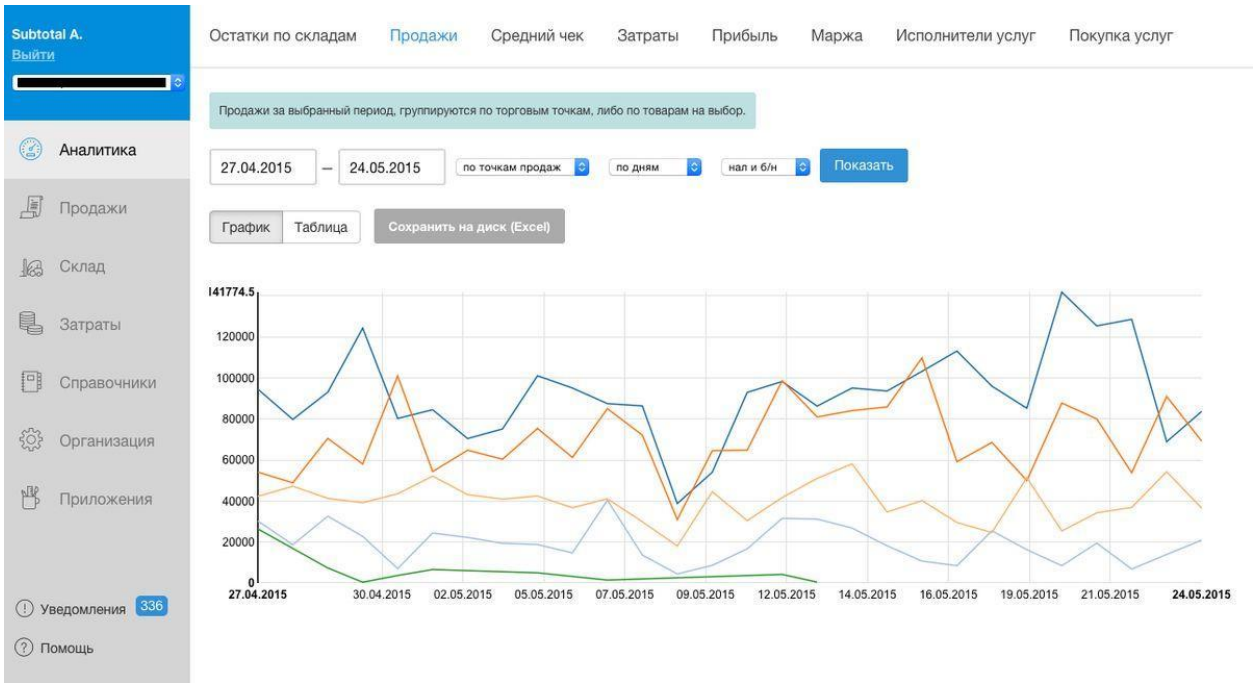

Рисунок 1.8 – Статистика «SUBTOTAL»

Qasl – це система для розумної автоматизації бізнесу. Основний продукт компанії – каса на планшеті і хмарний сервіс, що включає аналітику, CRM, склад і персонал. Напрямки діяльності – харчування, роздрібна торгівля та салони краси.

На рис. 1.9 зображено інтерфейс програми "QASL".

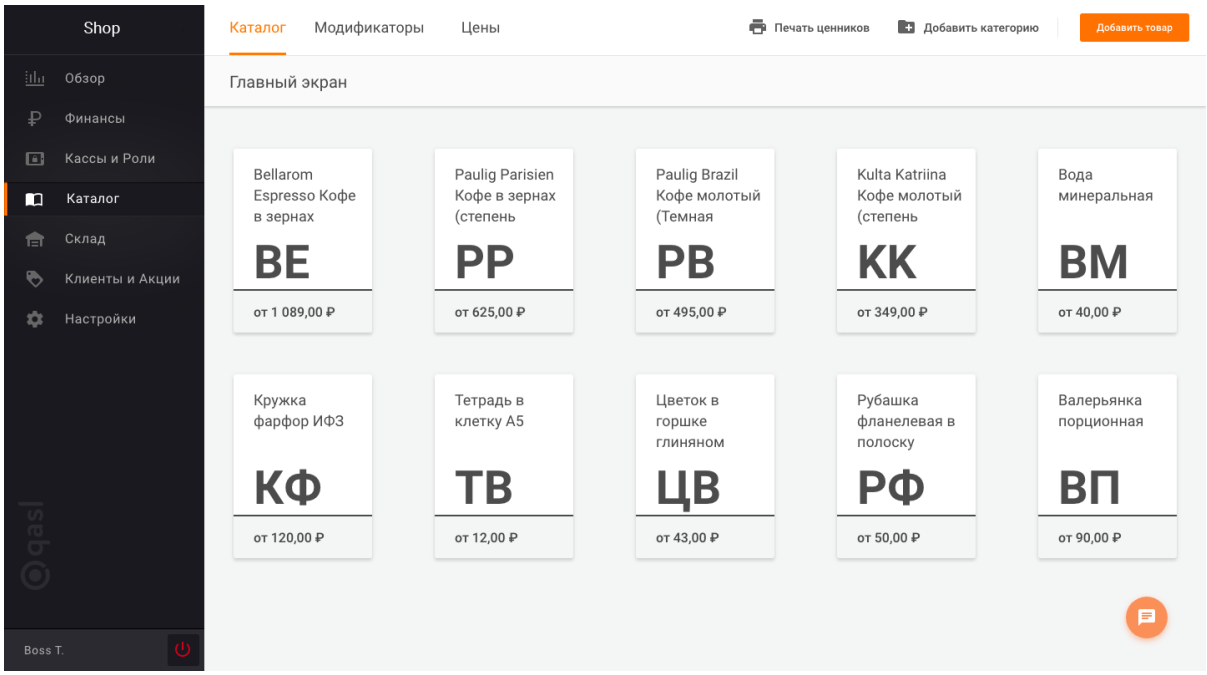

Рисунок 1.9 – Інтерфейс QASL

Переваги програми:

- програмою можна користуватися як на комп’ютерах, так і на мобільних гаджетах;
- підключення інтернет-магазину;
- доброзичливий, зрозумілий інтерфейс, що дозволяє провести первинне налаштування програми самостійно;
- доступні для малого бізнесу тарифні плани;
- наявність зручного модуля клієнтської бази;
- наявність повнофункціональної демо-версії, що дозволяє 14 днів користуватися всіма можливостями програми;
- підключення декількох магазинів і складів.

Недоліки програми:

- до програми підключається лише обмежений перелік моделей касового обладнання;
- відсутність телефонної підтримки, що помітно ускладнює вирішення технічних проблем;

- базовий тариф непотрібний, оскільки передбачає введення в базу тільки одного постачальника і одного покупця;
- відсутність аналітики за касирами і асортиментних груп;
- відсутня можливість доопрацювання програм під конкретного клієнта.

«1С: Торгівля і склад» призначена для обліку будь-яких видів торгових операцій. Завдяки гнучкості і можливості налаштування, система здатна виконувати всі функції обліку – від ведення довідників і введення первинних документів до отримання різних відомостей і аналітичних звітів.

Рис. 1.10 та 1.11 відображають інтерфейс програми “1С:Торгівля і склад”.

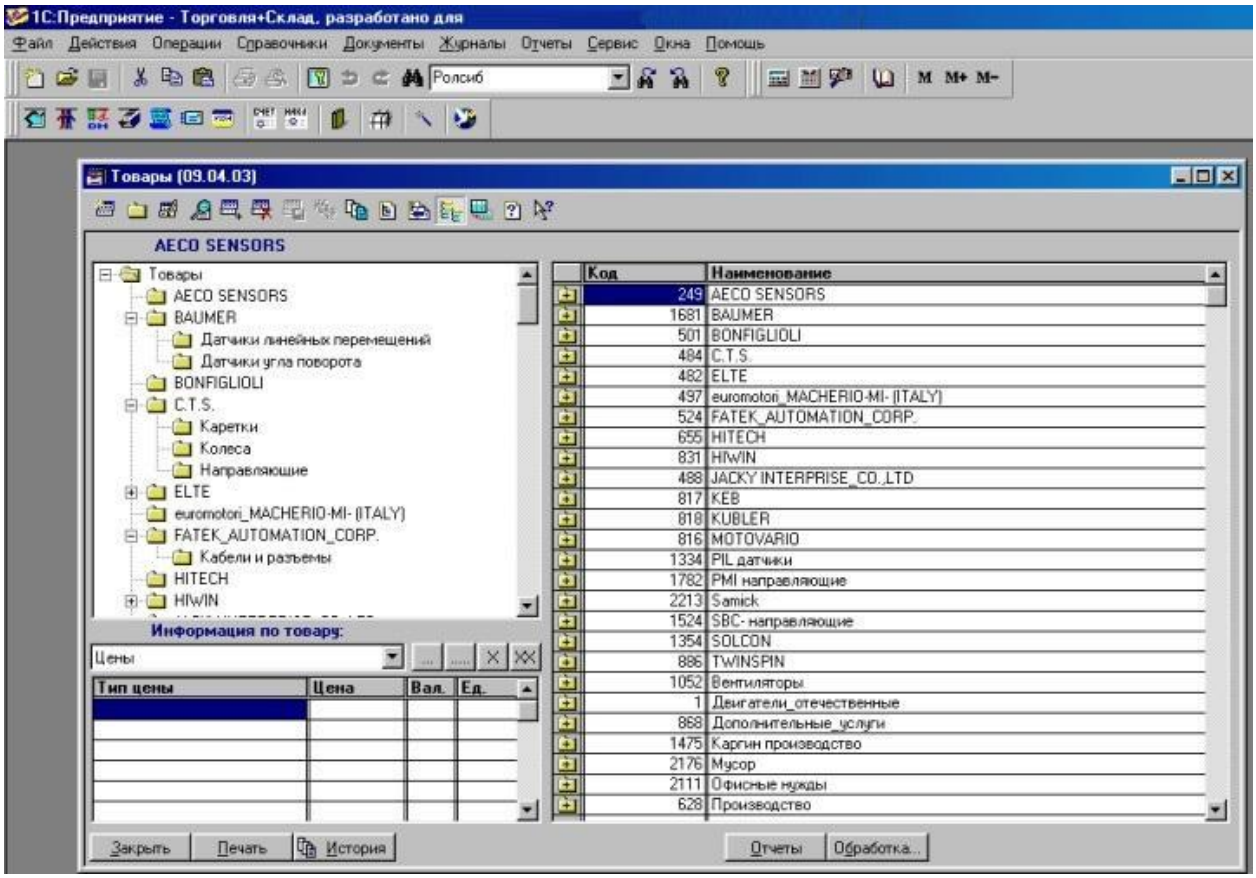

Рисунок 1.10 – Інтерфейс «Торгівля і склад»

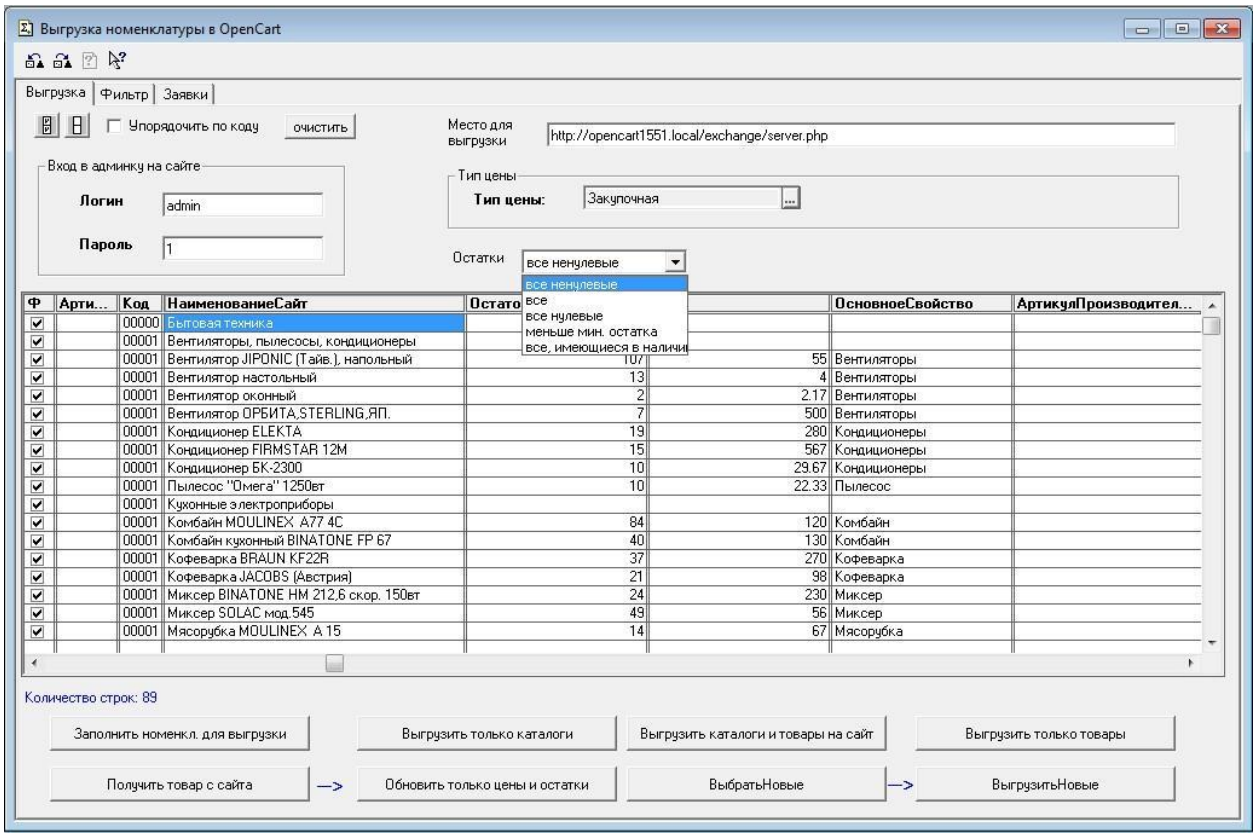

Рисунок 1.11 – Інтерфейс «Торгівля і склад»

Переваги програми «1С: Торгівля і склад»:

- наявність функціоналу для повноцінного бухгалтерського, податкового та складського обліку;
- інтеграція з будь-яким касовим і торговим устаткуванням;
- автоматичне формування всіх уніфікованих торговельних документів;
- консолідований облік в декількох торгових точках;
- висока стабільність роботи;
- можливість підстроювання меню і функціоналу під конкретного клієнта.

Недоліки програми «1С: Торгівля і склад»:

- висока вартість;
- складність навчання нових співробітників;
- тривалий період впровадження і налаштування;
- необхідність постійного оновлення ПЗ;
- відсутність CMR-системи;
- відсутність демо-версії.

Висновки до розділу 1

Отже, виходячи з вище перерахованих переваг та недоліків, можна зрозуміти, що інтерфейс застосунку, який має бути створений, повинен бути простим, а сам застосунок недорогим, так як метою є створення простого та доступного застосунку. Тому для створення візуальної частини варто використати Windows Form, який характеризується простими та зрозумілими інтерфейсами та мову програмування C#, тому що вона добре функціонує з базами даних.

РОЗДІЛ 2. АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Особливості програмної реалізації

Під час розробки програми було застосовану середовище розробки – Visual Studio, мову програмування – C#. за допомогою інтерфейсу програмування додатків Windows Form. За допомоги програмного забезпечення Adobe Photoshop було розроблено текстури для Button.

Рисунок 2.1 – Зображення логотипу Microsoft Visual Studio

Інтегроване середовище розробки Visual Studio є стартовою площадкою для написання, налагодження та зборки коду. Також за допомогою цієї системи можна публікувати застосунки. Інтегроване середовище (IDE) є багатофункціональною програмою, яку можна використати для різноманітних аспектів розробки програмного забезпечення. Окрім стандартного редактора, який існує у багатьох IDE, Visual Studio, включає в себе компілятори, засоби автозавершення коду, графічні коструктори і багато інших функцій для спрощення процесу розробки.

Рис. 2.2 зображає основний інтерфейс Visual Studio, який зустрічає користувача при запуску програми. На рис. 2.2 також зображені основні фрагменти інтерфейсу, а саме-редактор для написання коду та файловий менеджер, який дозволяє рухатися між різними файлами проекту під час розробки.

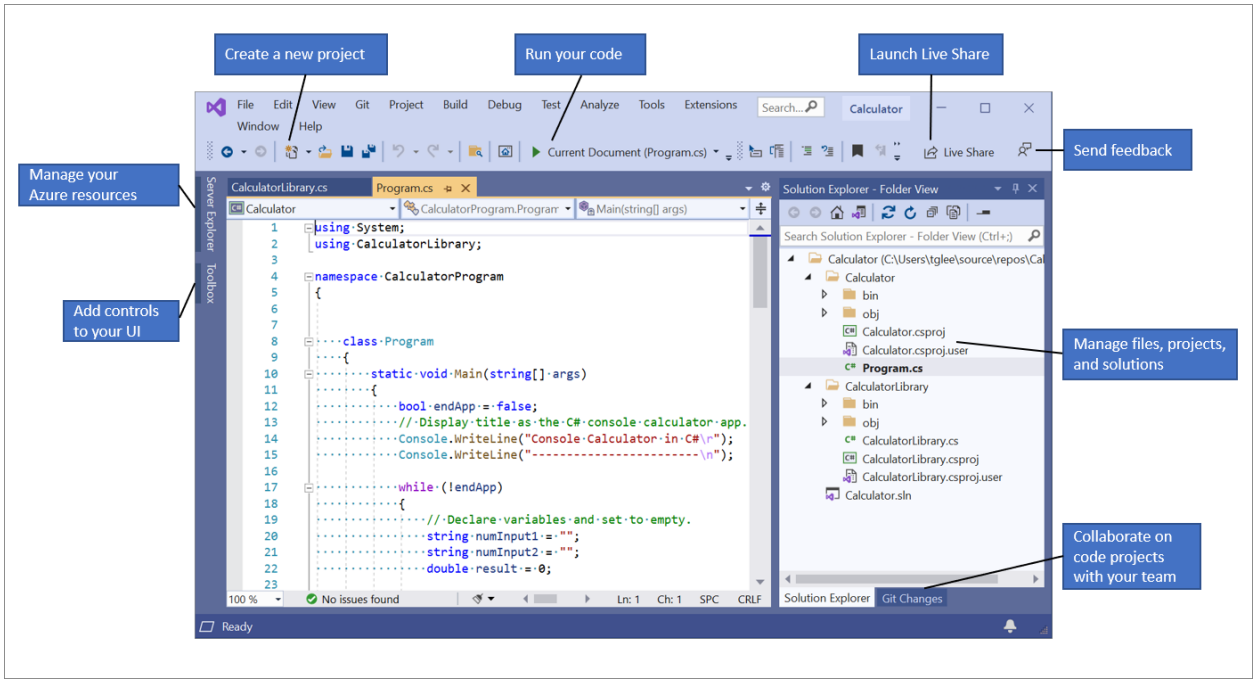

Рисунок 2.2 – Основний інтерфейс Visual Studio

Варто зазначити, що найактуальнішою версією Visual Studio є Visual Studio 19, яка поділяється на Community, Professional та Enterprise. Система доступна на Windows та Mac. Не дивлячись на це, деякі функції Visual Studio не доступні для користувачів Apple і потребують встановлення додаткових застосунків, плагінів та фреймворків для коректної роботи.

2.2 Мова програмування – C#

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

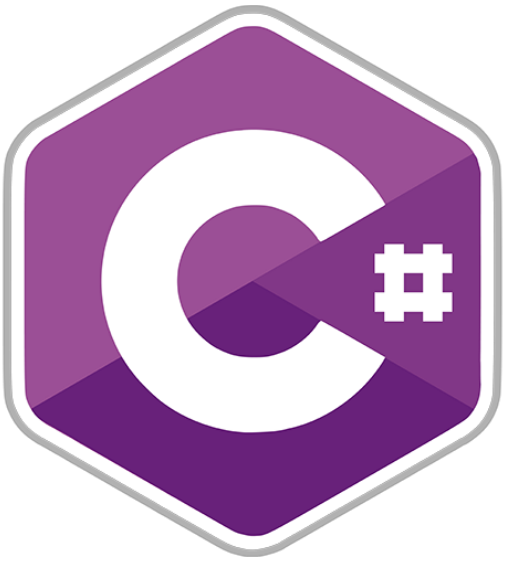

Рисунок 2.3 – Зображення логотипу мови програмування C#

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов C++, Object Pascal, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів.

2.3 Windows Form

Windows Forms — інтерфейс програмування застсунків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді. За допомогою Windows Forms та Visual Studio можна створювати інтлектуальні клієнтські застосунки, які відображають інформацію, запрошують ввід користувача і можуть контактувати з віддаленими комп’ютерами в мережі.

Рисунок 2.5 – Зображення логотипу інтерфейсу програмування застосунків Windows Form

В Windows Forms форма – це візуальна поверхня, на якій виводиться інформація для користувача. Зазвичай застосунки з Windows Forms будуються за допомогою додавання елементів взаємодії в форми для можливості введення даних та реагування на дії користувача.

Всередині .NET Framework, Windows Forms реалізується в межах простору імен System.Windows.Forms.

2.4 База даних – SQLite

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Рисунок 2.6 – Зображення логотипу базу даних SQLite

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Висновки до розділу 2

Програмне забезпечення було створене за допомогою вище перерахованих застосунків та систем. Основною задачою програми було облік даних та операціями типу додавання, віднімання, пошук, перенесення, вибірка за певним критерієм. Не менш важливим було розробити простий та зрозумілий кожному користувачу інтерфейс. Для виконання даного завдання було розроблено відповідні класи, методи перераховані вище та конкретно вказані у додатку А.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ЗАПАСАМИ

3.1 Вимоги до програми та структура застосунку

Інтерфейс користувача має містити:

- меню вибору складів (вибір здійснюється переміщенням курсору);
- меню вибраного складу(вибір здійснюється переміщенням курсору);
- форму для замовлення товару (вибір здійснюється переміщенням курсору);
- форму для реалізації товару (вибір здійснюється переміщенням курсору);
- форму для створення звіту (вибір здійснюється переміщенням курсору);
- форму для роботи з постачальниками(вибір здійснюється переміщенням курсору);
- поле введення даних (для виконання певної функції);
- поле відображення результатів;
- поле поточного стану таблиці (кількість записів, ім’я файлу тощо).

Програмна логіка має:

1. Забезпечити базові операції з даними:
   - додавання нового товару;
   - вилучення рядка по назві товару;
   - редагування товару;
   - додавання нового постачальника;
   - вилучення постачальника;
   - редагування постачальника;
   - створення звітностей
   - виведення всіх даних на екран.
2. Операції над даними:

- Продаж товару
- Пошук товару по унікальному коду.
- Вибірка товару згідно вибраного магазину
- Вибірка товару згідно типу товару
- Вибірка товару згідно ФІО покупця
- Вибірка товару згідно постачальнику товару
- Вибірка товару згідно дати прибуття товару або реалізації
- Вибірка постачальника згідно назви постальника
- Вибірка постачальника згідно ім’я контактної особи
- Вибірка постачальника згідно телефону постачальника
- Додавання постачальника
- Редагування даних постачальника
- Видалення постачальника

Рисунок 3.1 – Загальний алгоритм роботи програми

Данна програма містить вікно вибору складу, складу, замовлення товару, реалізації товару, створення звіту та постачальників.

Основні функції вікон:

Вікно вибору складу має функцію додавання складів до списку та відповідно список існуючих складів.

Вікно складу має функцію перегляду товарів які є на складі та є переходом до вікон замовлення товару, реалізації товару, створення звіту та постачальників.

Вікно замовлення товару має функцію замовлення вибраного товару, а саме вібир постачальника, товару та кількості товару.

Вікно реалізації товару має функцію реалізації товару, тобто таблицю з даними покупця, датою, товаром та кількістю.

Вікно створення звіту має функцію створення звіту за товаром, тобто звітності прибулого товару на склад, кількість, дата та час, а також реалізованого товару.

Вікно постачальників має функцію перегляду, додавання, редагування та видалення постачальників.

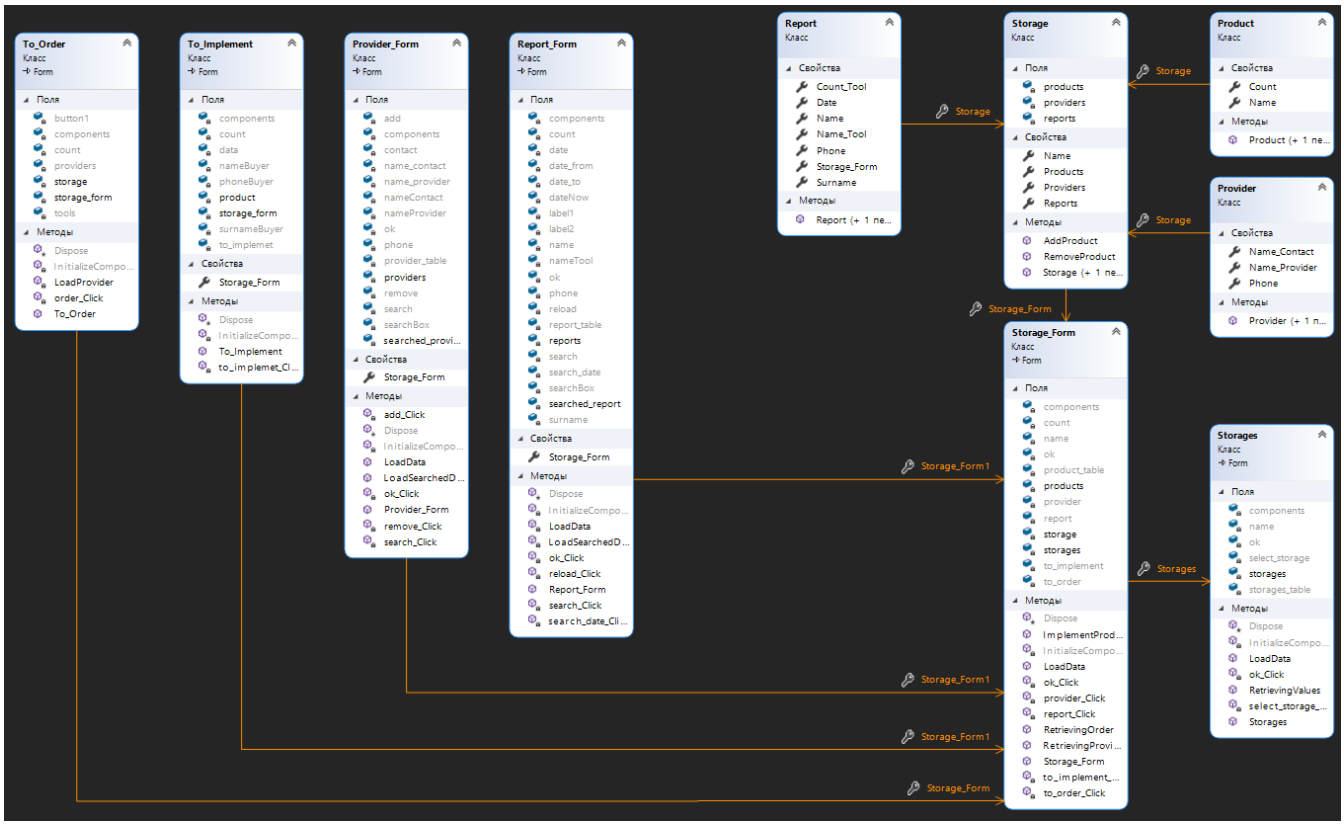

Рисунок 3.2 – Діаграма класів

Діаграма класів містить:

- Клас Product;
- Клас Provider;
- Клас Report;

- Клас Storage;
- Клас Storages;
- Клас Storage_Form;
- Клас Report_Form;
- Клас Report_Form;
- Клас To_Implement;
- Клас To_Order;

Клас Storage – клас описує об’єкт склад. Має містити в собі данні про товар який знаходиться на складі, звіти по товару та дані про постачальників тому має наступні поля:

- List<Report> reports – поле яке несе інформацію про звіти на складі;
- List<Product> products – поле яке несе інформацію про товар на складі;
- List<Provider> providers – поле яке несе інформацію про постачальників;
- Name – поле яке несе інформацію про назву складу;

Має основні 2 методи:

- AddProduct – додавання товару на склад;
- RemoveProduct – видалення товару зі складу;

Клас Product – клас, що описує об’єкт товар. Має містити в собі дані про товар, а саме назву та кількість товару тому має наступні поля:

- Name – поле яке несе інформацію про назву товару;
- Count – поле яке несе інформацію про кількість товару;

Клас Report – клас, що описує об’єкт звіт. Має містити в собі дані про звіт, а саме ім’я, прізвище та номер телефону покупця, дата реалізації товару, назва та кількість товару тому має наступні поля:

- Name – поле яке несе інформацію про ім’я покупця;
- Surname – поле яке несе інформацію про прізвище покупця;
- Phone – поле яке несе інформацію про номер телефону покупця;
- Date – поле яке несе інформацію про дату реалізації товару;
- Name_Tool – поле яке несе інформацію про назву реалізованого товару;
- Count_Tool – поле яке несе інформацію про кількість реалізованого товару;

Клас Provider – клас, що описує об’єкт постачальника. Має містити в собі дані про назву, ФІО та номер телефону контактної особи тому має наступні поля:

- Name_Contact – поле яке несе інформацію про ФІО контактної особи;
- Name_Provider – поле яке несе інформацію про назву постачальника;
- Phone – поле яке несе інформацію про номер телефону особи.

3.2 Структура бази даних

В програмному застосунку було застосовано використання бази даних SQLite. Вибір був зроблений на ній тому, що дана база даних має менший функціонал, що сприяє зменшеності часу на розгортання, та не потребує ніяких вмінь та навичок від користувача.

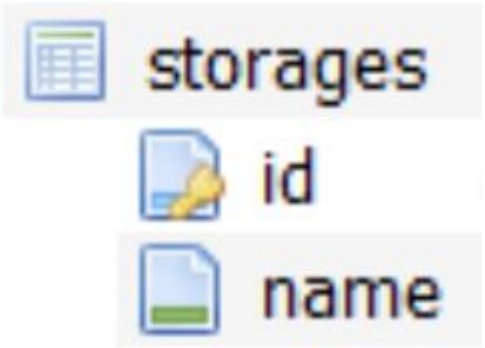

Рис 2.2 Таблиця з назвами складів

Має поля:

- id – ідентифікатор складу;
- name – назва складу;

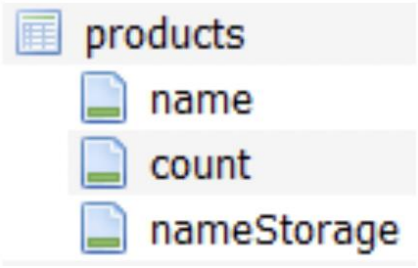

Рисунок 3.3 – Таблиця з продуктами

Має поля:

- name – назва продукту;
- count – кількість продуктції;
- nameStorage – назва складу на якому знаходиться продукт;

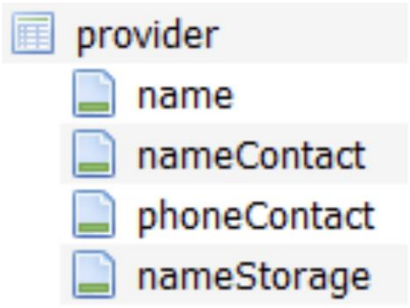

Рисунок 3.4 – Таблиця з постачальниками

Має поля:

- name – назва постачальника;
- nameContact – ім’я контактної особи;
- phoneContact – номер телефону контактної особи;
- nameStorage – назва складу на яку постачає данний постачальник;

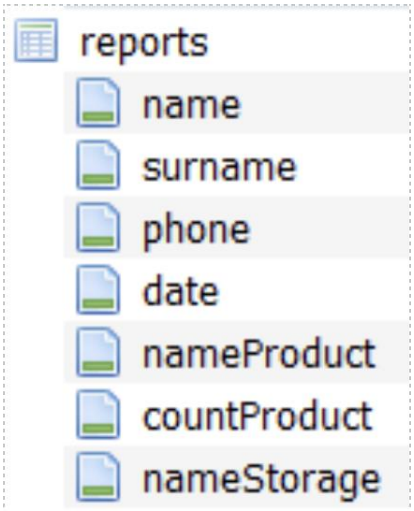

Рисунок 3.5 – Таблиця із звітами

Має поля:

- name – ім’я покупця;
- surname – прізвище покупця;
- phone – телефон покупця;
- date – дата реалізації;
- nameProduct – назва товару;
- countProduct – кількість товару;
- nameStorage – назва складу на якому зроблено звіт;

3.3 Порядок використання програмного застосунку

На рис. 3.6 зображено вікно зі списком існуючих складів та їх назвами. Кожен склад має товари, звіти та постачальників.

Для входження в меню складу потрібно вибрати певний склад на натиснути кнопку “Увійти” (рис. 3.7).

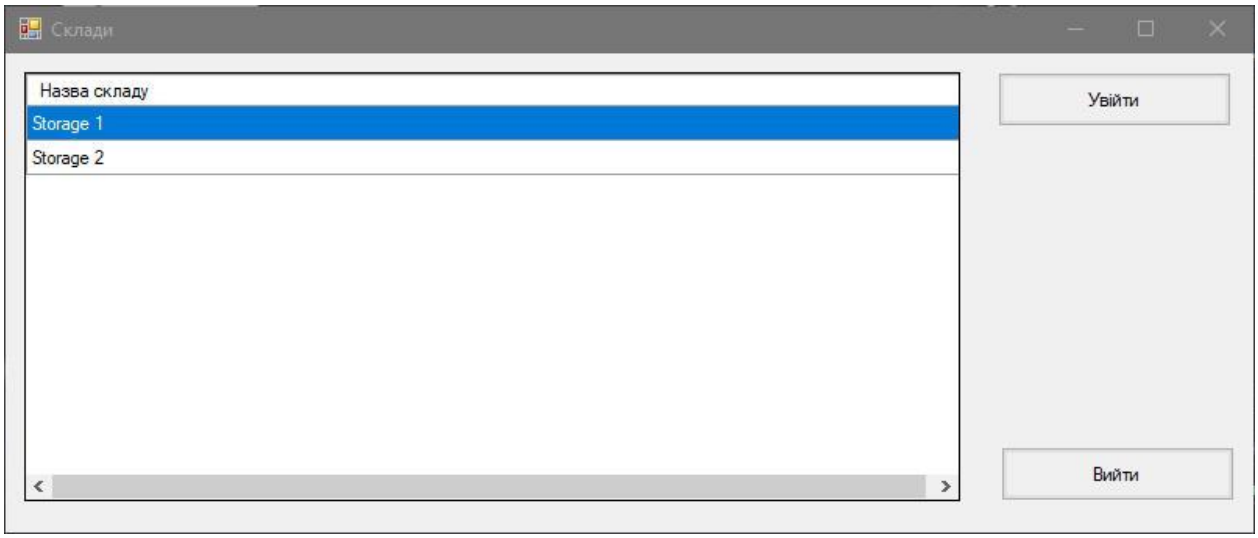

Рисунок 3.6 – Меню існуючих складів

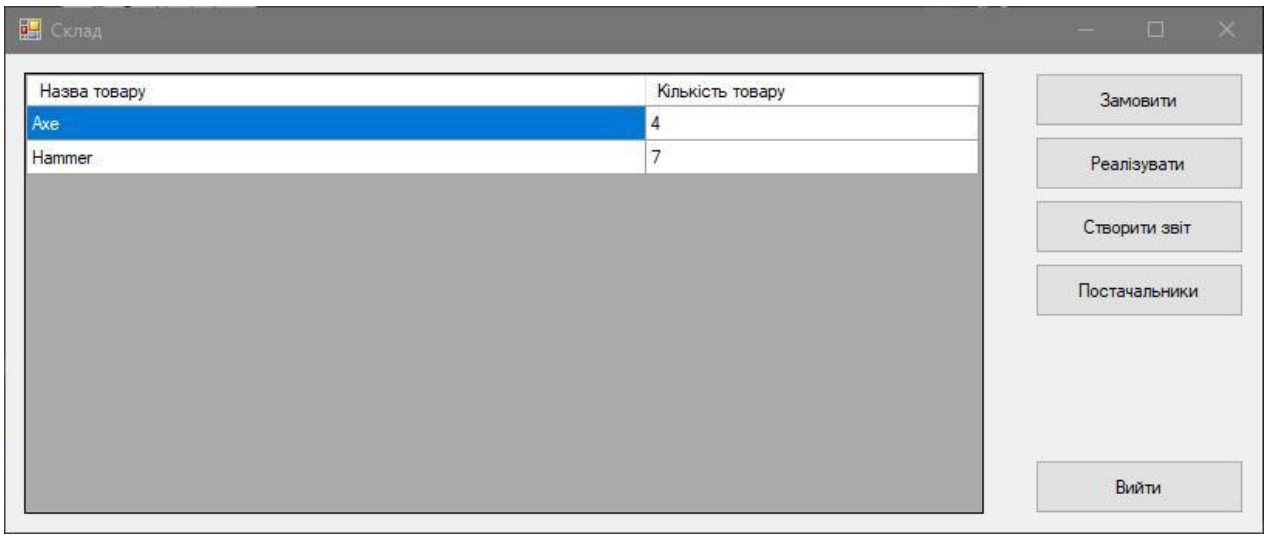

Рисунок 3.7 – Меню складу

Меню складу має список товару який складається з таких параметрів як “Назва товару” та “Кількість товару” (рис. 3.8). Кнопка “Замовити” слугує для відкриття вікна замовлень. Кнопка “Реалізувати” слугує для відкриття вікна реалізації товару. Кнопка “Створити звіт” слугує для відкриття вікна звітів. Кнопка “Постачальники” слугує для відкриття вікна постачальників.

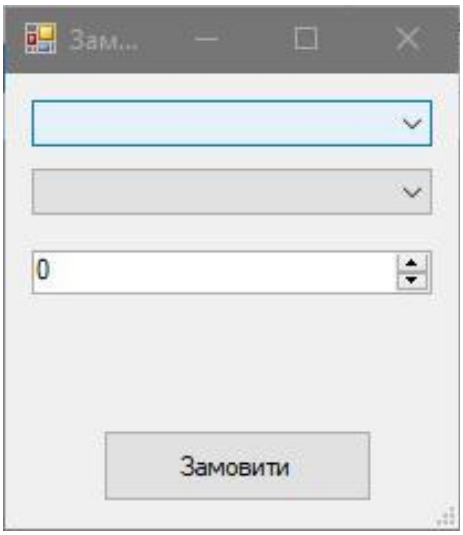

Рисунок 3.8 – Меню замовлення товару

Меню замовлення товару слугує для замовлення товару. Для замовлення товару потрібно обрати:

- Постачальника
- Товар
- Кількість товару

Товар буде відображено на складі, а у звіті буде створено запис з назвою постачальника, товар, кількість товару та дату прибуття товару на склад.

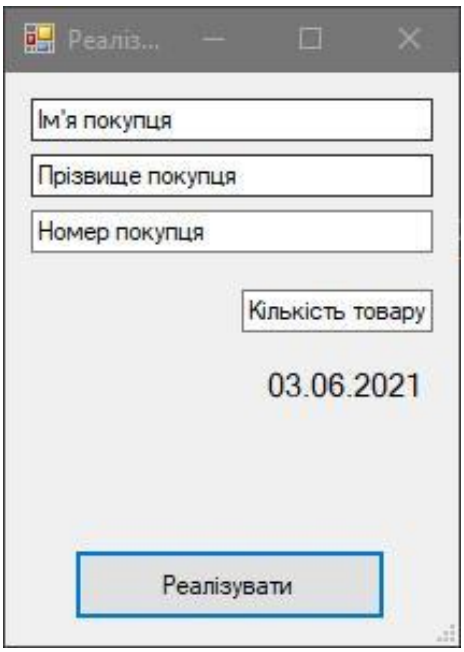

Рисунок 3.9 – Меню реалізації товару

Вікно реалізації товару слугує для реалізації вибраного товару у вікні “Склад”. Для реалізації товару користувачу потрібно вказати:

- Ім’я покупця
- Прізвище покупця
- Номер мобільного телефону покупця
- Кількість реалізації товару

Товар буде відображено на складі, а у звіті буде створено запис з ім’ям, прізвищем, номер покупця, датою реалізації, назвою та кількістю товару.

Меню створення звітності (рис. 3.10) слугує для створення звітності по складу з функціями вибірки за:

- Ім’ям
- Прізвищем
- Номером телефону
- Датою
- Назвою товару
- Кількістю товару
- Проміжком дат реалізації товару

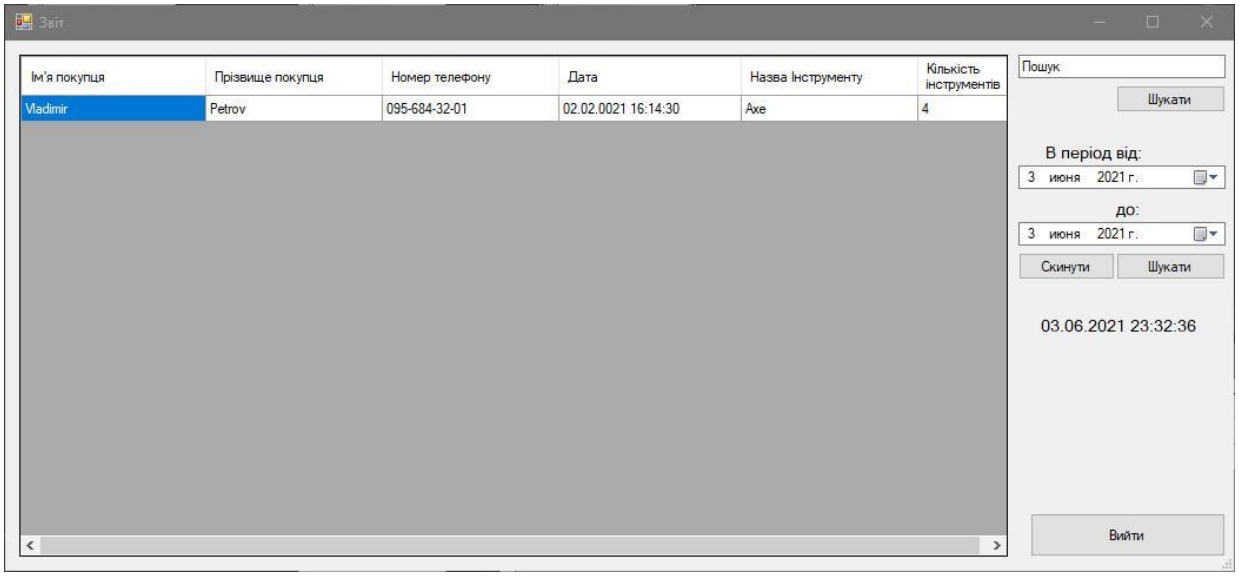

Рисунок 3.10 – Меню створеннння звітності

Меню постачальників (рис. 3.11) слугує для перегляду, додавання, редагування та видалення постачальників, а також для пошуку постачальників за такими параметрами як:

- Назва постачальника
- Ім’я контактної особи
- Номер телефону контактної особи

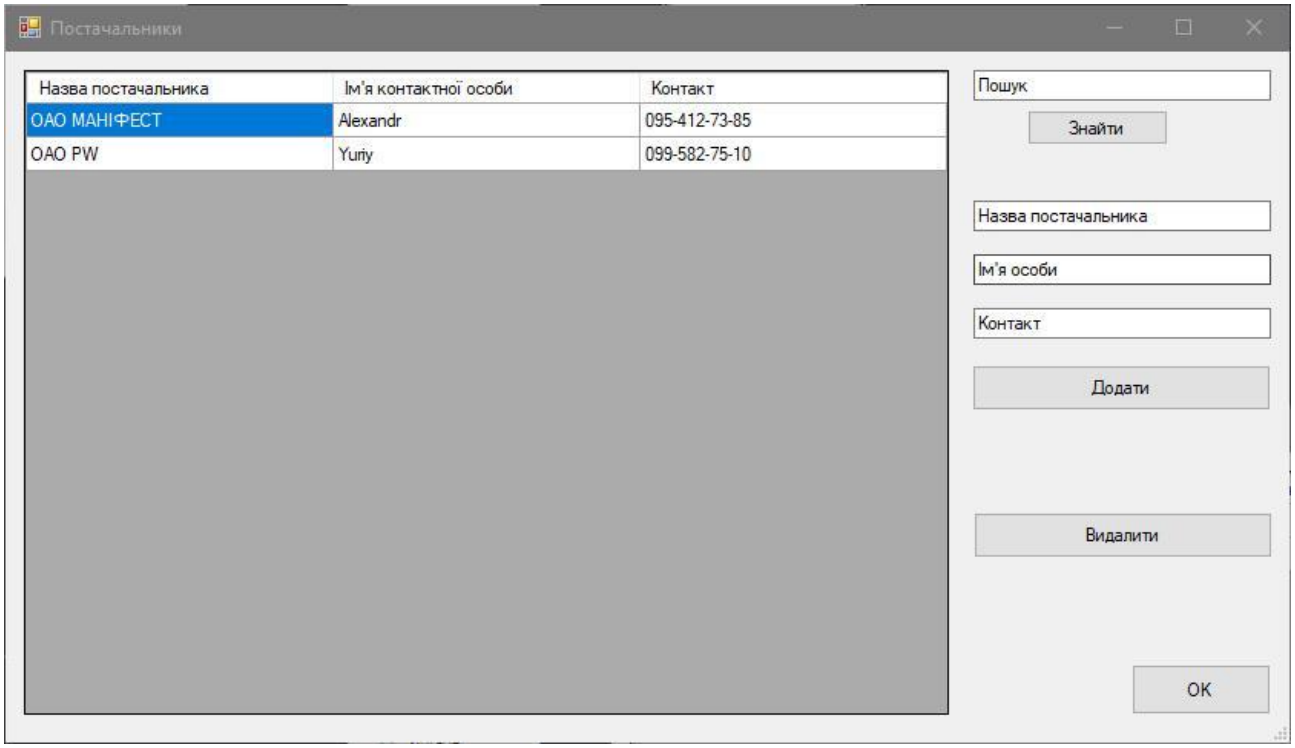

Рисунок 3.11 – Меню постачальників

Висновки до розділу 3

У цьому розділі, ми показали, як працювати з програмою для користувача, для того, щоб розробити інтерфейс була використана інтерфейсу програмування додатків Windows Form та мова програмування C#. Інтерфейс є легким та інтуїтивно зрозумілим. Мова C# та платформа Windows Form надають додатку невибагливості до технічного забезпечення користувача та швидкості обробки даних.

ВИСНОВКИ

Для розробки оптимізованої системи було проведено дослідження сучасних застосунків, що дозволяють автоматизовано вести облік товарів. Після досліджень було здійснено порівняння самих популярних застосунків та виявлено ряд недоліків які є характерними для них та поставлено технічне завдання з розробки програмного додатку націлене на усування даних недоліків та створення програмного забезпечення максимально простого для технічного забезпечення та простого у використані звичайним користувачем.

Для розробки програмного застосунку було використано найбільш популярне середовище розробки Microsoft Visual Studio.

В даній роботі приведені результати досліджень найпопулярніших CMS систем, програмне супроводження веб-застосунка та детальний опис встановлення та налаштування розробленого шаблона, функцій, компонентів та модулів.

За останні роки відбувся різкий стрибок у розвитку комп’ютерної техніки й програмного забезпечення із одночасним розширенням сфер застосування персональних комп’ютерів. Галузь розробки комп’ютерних програм є важливою складовою науково-технологічного прогресу.

По-перше, з поставленого завдання випливає, що для майбутнього застосунку необхідно використовувати базу даних, задля забезпечення подальшої зручної розробки програми та кращого розуміння написаного коду.

По-друге, з поставленого завдання випливає, що потрібно розробити таблиці даних, що містить певні поля заданої назви та типу даних, які постійно зберігаються у базі даних, а також забезпечити базові операції з даними (додавання нового рядка; вилучення рядка по коду; редагування; виведення всіх даних на екран). По-третє, потрібно було розробити покрокові інструкції основних та додаткових операцій. Загалом, інтерфейс програми є інтуїтивно зрозумілим та не потребує додаткових пояснень чи уточнень та не потребує спеціальних навичок для користування програмою.

Отже, завдання кваліфікаційної роботи виконано, мету роботи досягнуто, а зокрема: розроблено графічний програмний застосунок, що не вимагає додаткового навчання для використання програми.

В кваліфікаційній роботі проведено аналіз предметної області, існуючих варіантів розв’язання досліджуваної задачі. Проведено порівнальний аналіз аналогічних систем. Був проведений аналіз методів розв’язання задачі.

В роботі було досліджено та проаналізовано найбільш важливу і актуальну інформацію щодо розробки системи управління товарами на складі а саме основні принципи створення бази даних, їх структуру і функціональність, взаємодію основних компонентів. Також були розглянуті новітні та найбільш перспективні технології, які з успіхом вже використовуються користувачами по всьому світі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Використання веб-ресурсів для покращення візуального сприйняття інформації [електронний ресурс] // Режим доступу: http http://inmad.vntu.edu.ua/portal/index.php – Назва з екрану.

2. Використання веб-ресурсів [електронний ресурс] // Режим доступу: http://galanet.at.ua/publ/1-1-0-23– Назва з екрану.

3. Етапи створення веб-ресурсів [електронний ресурс] // Режим доступу: http://edufuture.biz/index.php?title=Етапи створення веб- ресурсів – Назва з екрану.

4. 5plus.com [електронний ресурс] // Режим доступу: http://www.5plus.com/ru – Назва з екрану.

5. Teacher.com [електронний ресурс] // Режим доступу: http://www.teacher.com/ – Назва з екрану.

6. MyClass.com [електронний ресурс] // Режим доступу: http://myclass.com/ – Назва з екрану.

7. Козловський В.О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах [Навчальний посібник]. Вінниця: ВДТУ, 2003. 73 с.

8. Аналіз попиту [електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/ Аналіз_попиту – Назва з екрану.

9. HTML 5 [електронний ресурс]// Режим доступу: https://ru.wikipedia.org/wiki/HTML5 - Назва з екрану

10. PHP 5 / Д. В. Котеров, А. Ф. Костарев. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008; Зельдман Д.

11. Bootstrap [електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Bootstrap - Назва з екрану

12. Java Script [електронний ресурс] // Режим доступу: https://ru.wikipedia.org/wiki/javascript - Назва з екрану

13. MySQL [електронний ресурс] // Режим доступу: https://www.mysql.com/

- Назва з екрану

14. Oracle Database[електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Oracle_DB - Назва з екрану

15. Aptana Studio [електронний ресурс] // Режим доступу: http://www.aptana.com/ – Назва з екрану.

16. Php designer [електронний ресурс // Режим доступу: http://www.phpdesigner.com/ – Назва з екрану.

17. Sublime Text 3 [електронний ресурс] // Режим доступу: http://eax.me/sublime-text/ – Назва з екрану.

18. Романюк О.Н. Веб-дизайн і комп'ютерна графіка. Навчальний посібник/ О.Н. Романюк, Д.І. Катєльніков, О.П. Косовець. Вінниця: ВНТУ, 2007. 147 с.

19. ER-модель даних [електронний ресурс] // Режим доступу: http://uk.wikipedia.org/wiki/ER_модель_даних – Назва з екрану

20. Зельдман Д. Web-дизайн по стандартам / Д. Зельдман. Москва: НТ Пресс, 2005. 440 с.

21. Adobe Flash Professional CS5 [електронний ресурс] // Режим доступу: http://getintopc.com/softwares/development/adobe-flash- professional-cs5/ – Назва з екрану.

22. ActionScript 2.0 [електронний ресурс] // Режим доступу: https://ru.wikipedia.org/wiki/ActionScript - Назва з екрану

23. ActionScript 3.0 [електронний ресурс] // Режим доступу: http://help.adobe.com/ru_RU/ActionScript/3.0 - Назва з екрану

24. Иттен Иоханнес. Искусство цвета / Иоханнес Иттен. изд. Д. Аронов, 2011. 96с. ISBN 978-5-94056-021-0

25. Айзенберг Брайан. Тестирование и оптимизация веб-сайтов: руководство по Google Website Optimizer / Брайан Айзенберг, Джон Кварто вон Тивадар, Лайза Т. Дэвис. изд. Диалектика, 2009. 336 с. ISBN 978- 5-8459-1542-9

26. Тестування методом чорного ящика [електронний ресурс] // Режим доступу: http://ru.qatestlab.com/services/no-documentation/black-box-testing/ – Назва з екрану

27. Кошторис витрат виробництва: поняття, склад та методика складання [електронний ресурс] // Режим доступу:

28. http://pidruchniki.com/1373112064744/ekonomika/koshtoris_vitrat_virobnitstva_p onyattya_sklad_metodika_skladannya – Назва з екрану.

ДОДАТКИ

ДОДАТОК А

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
	public partial class Storages : Form
	{
		private List<Storage> storages = new List<Storage>()
		{
			new Storage("Storage 1", new List<Product>(){
				new Product("Axe", 4),
				new Product("Hammer", 7)}, new List<Report>(){
				new Report("Vladimir", "Petrov", "095-684-32-01", new DateTime(21, 02, 02, 16, 14, 30), "Axe", 4)},
				new List<Provider>() {
					new Provider("ОАО МАНІФЕСТ", "Alexandr", "095-412-73-85"),
					new Provider("ОАО PW", "Yuriy", "099-582-75-10")
				}),
			new Storage("Storage 2", new List<Product>(){
				new Product("Tape", 7),
				new Product("Hammer", 10)}, new List<Report>(){
				new Report("Vladislav", "Vadimovich", "097-124-42-85", new DateTime(21, 05, 02, 09, 49, 27), "Axe", 4)},
				new List<Provider>() {
					new Provider("ОАО TOOLS", "Vladislav", "099-441-51-82"),
					new Provider("ОАО IO", "Yuriy", "063-752-62-23")}
				)};

		public Storages()
		{
			InitializeComponent();
			LoadData();
		}

		public void LoadData()
		{
			foreach (Storage storage in storages)
			{
				storages_table.Rows.Add(storage.Name);
			}
		}

		private void select_storage_Click(object sender, EventArgs e)
		{
			foreach (Storage storage in storages)
			{
				if (storage.Name == storages_table.CurrentCell.Value.ToString())
				{
					Storage_Form storage_Form = new Storage_Form(storage, this);
					storages.Remove(storage);
					storage_Form.Show();
					this.Hide();
					break;
				}
			}
		}

		public void RetrievingValues(Storage storage)
		{
			storages.Add(storage);
		}

		private void ok_Click(object sender, EventArgs e)
		{
			this.Close();
		}
	}
}
```

```
using iTextSharp.text.pdf;
using NLipsum.Core;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
	public partial class Storage_Form : Form
	{
		List<Product> products = new List<Product>();
		Storages storages = new Storages();
		Storage storage = new Storage();

		public Storage_Form(Storage storage, Storages storages)
		{
			InitializeComponent();
			products = storage.Products;
			this.storage = storage;
			this.storages = storages;
			LoadData();
		}

		public Storages Storages
		{
			get => default;
			set
			{
			}
		}

		public void LoadData()
		{
			product_table.Rows.Clear();

			foreach (Product product in products)
			{
				product_table.Rows.Add(product.Name, product.Count);
			}
		}

		private void ok_Click(object sender, EventArgs e)
		{
			storages.RetrievingValues(new Storage(storage.Name, products, storage.Reports, storage.Providers));
			storages.Show();
			this.Close();
		}

		private void to_order_Click(object sender, EventArgs e)
		{
			To_Order to_order = new To_Order(this, storage);
			to_order.ShowDialog();
		}

		public void RetrievingOrder(Report report)
		{
			foreach (Product temp_product in products)
			{
				if (temp_product.Name == report.Name_Tool)
				{
					products.Remove(temp_product);
					products.Add(new Product(temp_product.Name, temp_product.Count + report.Count_Tool));
					storage.Reports.Add(report);
					return;
				}
			}

			products.Add(new Product(report.Name_Tool, report.Count_Tool));
			storage.Reports.Add(report);
		}

		private void to_implement_Click(object sender, EventArgs e)
		{
			foreach (Product temp_product in products)
			{
				if (temp_product.Name == product_table.CurrentCell.Value.ToString())
				{
```

```
                                        To_Implement to_Implement = new To_Implement(this, temp_product);
                                        to_Implement.ShowDialog();
                                        return;
                                }
                        }
                }

                public void ImplementProduct(Report report)
                {
                        foreach (Product temp_product in products)
                        {
                                if (temp_product.Name == report.Name_Tool)
                                {
                                        products.Remove(temp_product);
                                        products.Add(new Product(temp_product.Name, temp_product.Count - report.Count_Tool));
                                        storage.Reports.Add(report);
                                        return;
                                }
                        }

                        products.Add(new Product(report.Name, report.Count_Tool));
                        storage.Reports.Add(report);
                }

                private void report_Click(object sender, EventArgs e)
                {
                        Report_Form report_Form = new Report_Form(storage.Reports);
                        report_Form.ShowDialog();
                }

                private void provider_Click(object sender, EventArgs e)
                {
                        Provider_Form provider_Form = new Provider_Form(storage.Providers);
                        provider_Form.ShowDialog();
                }

                public void RetrievingProvider(List<Provider> provider)
                {
                        storage.Providers = provider;
                }
        }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
        public class Storage
        {
                private List<Product> products;
                private List<Report> reports;
                private List<Provider> providers;

                public Storage() { }
                public Storage(string name, List<Product> products, List<Report> reports, List<Provider> providers)
                {
                        Name = name;
                        Products = products;
                        Reports = reports;
                        Providers = providers;
                }

                public string Name { get; set; }
                public List<Product> Products {
                        get
                        {
                                return products;
                        }
                        set
                        {
                                products = value;
                        }
                }

                public List<Report> Reports
                {
                        get
                        {
                                return reports;
```

```
                        }
                        set
                        {
                                reports = value;
                        }
                }

                public List<Provider> Providers
                {
                        get
                        {
                                return providers;
                        }
                        set
                        {
                                providers = value;
                        }
                }

                public Storage_Form Storage_Form
                {
                        get => default;
                        set
                        {
                        }
                }

                public void AddProduct(Product product)
                {
                        products.Add(product);
                }

                public void RemoveProduct(int index)
                {
                        products.RemoveAt(index);
                }
        }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
        public class Product
        {
                public Product() { }
                public Product(string name, int count)
                {
                        Name = name;
                        Count = count;
                }

                public string Name { get; set; }
                public int Count { get; set; }

                public Storage Storage
                {
                        get => default;
                        set
                        {
                        }
                }
        }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
	public class Provider
	{
		public Provider() { }
		public Provider(string name_provider, string name_contact, string phone)
		{
			Name_Provider = name_provider;
			Name_Contact = name_contact;
			Phone = phone;
		}

		public string Name_Provider { get; set; }
		public string Name_Contact { get; set; }
		public string Phone { get; set; }

		public Storage Storage
		{
			get => default;
			set
			{
			}
		}
	}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
	public class Report
	{
		public Report() { }
		public Report(string name, string surname, string phone, DateTime date, string name_tool ,int count_tool)
		{
			Name = name;
			Surname = surname;
			Phone = phone;
			Date = date;
			Name_Tool = name_tool;
			Count_Tool = count_tool;
		}

		public string Name { get; set; }
		public string Surname { get; set; }
		public string Phone { get; set; }
		public DateTime Date { get; set; }
		public string Name_Tool { get; set; }
		public int Count_Tool { get; set; }

		public Storage_Form Storage_Form
		{
			get => default;
			set
			{
			}
		}

		public Storage Storage
		{
			get => default;
			set
			{
			}
		}
	}
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
	public partial class Report_Form : Form
	{
		List<Report> reports;
		List<Report> searched_report = new List<Report>();
		public Report_Form(List<Report> reports)
		{
			InitializeComponent();
			dateNow.Text = DateTime.Now.ToString();
			this.reports = reports;
			LoadData();
		}

		public Storage_Form Storage_Form
		{
			get => default;
			set
			{
			}
		}

		public Storage_Form Storage_Form1
		{
			get => default;
			set
			{
			}
		}

		private void LoadData()
		{
			report_table.Rows.Clear();

			foreach (Report report in reports)
			{
				report_table.Rows.Add(report.Name, report.Surname, report.Phone, report.Date.ToString(), report.Name_Tool,
report.Count_Tool);
			}
		}

		private void LoadSearchedData()
		{
			report_table.Rows.Clear();

			foreach (Report report in searched_report)
			{
				report_table.Rows.Add(report.Name, report.Surname, report.Phone, report.Date, report.Name_Tool,
report.Count_Tool);
			}
		}

		private void ok_Click(object sender, EventArgs e)
		{
			this.Close();
		}

		private void search_Click(object sender, EventArgs e)
		{
			foreach (Report report in reports)
			{
				if ((searchBox.Text == report.Name) || (searchBox.Text == report.Surname)
					|| (searchBox.Text == report.Phone) || (searchBox.Text == Convert.ToString(report.Date))
						|| (searchBox.Text == report.Name_Tool) || (searchBox.Text ==
Convert.ToString(report.Count_Tool)))
				{
					searched_report.Add(report);
				}
				else if (searchBox.Text == "")
				{
					searched_report.Clear();
					LoadData();
					return;
				}
```

```
                }

                LoadSearchedData();
            }

            private void search_date_Click(object sender, EventArgs e)
            {
                foreach (Report report in reports)
                {
                    if ((report.Date > date_from.Value) && (report.Date < date_to.Value))
                    {
                        searched_report.Add(report);
                    }
                }

                LoadSearchedData();
            }

            private void reload_Click(object sender, EventArgs e)
            {
                searched_report.Clear();
                LoadData();
            }
        }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
    public partial class Provider_Form : Form
    {
        List<Provider> providers = new List<Provider>();
        List<Provider> searched_providers = new List<Provider>();
        public Provider_Form(List<Provider> providers)
        {
            InitializeComponent();
            this.providers = providers;
            LoadData();
        }

        public Storage_Form Storage_Form
        {
            get => default;
            set
            {
            }
        }

        public Storage_Form Storage_Form1
        {
            get => default;
            set
            {
            }
        }

        public void LoadData()
        {
            provider_table.Rows.Clear();

            foreach (Provider provider in providers)
            {
                provider_table.Rows.Add(provider.Name_Provider, provider.Name_Contact, provider.Phone);
            }
        }

        public void LoadSearchedData()
        {
            provider_table.Rows.Clear();

            foreach (Provider provider in searched_providers)
            {
                provider_table.Rows.Add(provider.Name_Provider, provider.Name_Contact, provider.Phone);
            }
        }
```

```
                private void ok_Click(object sender, EventArgs e)
                {
                        this.Close();
                }

                private void remove_Click(object sender, EventArgs e)
                {
                        foreach (Provider provider in providers)
                        {
                                if (provider.Name_Provider == provider_table.CurrentCell.Value.ToString())
                                {
                                        providers.Remove(provider);
                                        break;
                                }
                        }

                        LoadData();
                }

                private void search_Click(object sender, EventArgs e)
                {
                        foreach (Provider provider in providers)
                        {
                                if ((provider.Name_Provider == searchBox.Text) || (provider.Name_Contact == searchBox.Text)
                                                                                        || (provider.Phone == searchBox.Text))
                                {
                                        searched_providers.Add(provider);
                                }
                                else if (searchBox.Text == "")
                                {
                                        searched_providers.Clear();
                                        LoadData();
                                        return;
                                }
                        }

                        LoadSearchedData();
                }

                private void add_Click(object sender, EventArgs e)
                {
                        if ((name_provider.Text == "") || (name_contact.Text == "") || (contact.Text == ""))
                        {
                                MessageBox.Show("Невірно введенні данні");
                                return;
                        }

                        providers.Add(new Provider(name_provider.Text, name_contact.Text, contact.Text));
                        LoadData();
                }
        }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
        public partial class To_Implement : Form
        {
                Storage_Form storage_form;
                Product product;
                public To_Implement(Storage_Form storage_form, Product product)
                {
                        InitializeComponent();
                        this.storage_form = storage_form;
                        this.product = product;
                        data.Text = DateTime.Now.ToString();
                }

                public Storage_Form Storage_Form
                {
                        get => default;
                        set
                        {
                        }
```

```
                }

                private void to_implemet_Click(object sender, EventArgs e)
                {
                        Report report = new Report(nameBuyer.Text, surnameBuyer.Text, phoneBuyer.Text, DateTime.Now, product.Name,
int.Parse(count.Text));
                        storage_form.ImplementProduct(report);
                        storage_form.LoadData();
                        this.Close();
                }
        }
}


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom
{
        public partial class To_Order : Form
        {
                Storage_Form storage_form;
                Storage storage;
                public To_Order(Storage_Form storage_form, Storage storage)
                {
                        InitializeComponent();
                        this.storage_form = storage_form;
                        this.storage = storage;
                        LoadProvider();
                }

                public Storage_Form Storage_Form
                {
                        get => default;
                        set
                        {
                        }
                }

                private void LoadProvider()
                {
                        foreach (Provider provider in storage.Providers)
                        {
                                providers.Items.Add(provider.Name_Provider);
                        }
                }

                private void order_Click(object sender, EventArgs e)
                {
                        Provider provider = new Provider();

                        if (providers.SelectedItem == null)
                        {
                                MessageBox.Show("Виберіть постачальника!");
                                return;
                        }
                        else
                        {
                                foreach (Provider temp_provider in storage.Providers)
                                {
                                        if (temp_provider.Name_Provider == providers.SelectedItem.ToString())
                                        {
                                                provider = temp_provider;
                                        }
                                }
                        }


                        Report report = new Report("Прибуло на склад", provider.Name_Provider, provider.Phone, DateTime.Now,
tools.SelectedItem.ToString(), Convert.ToInt32(count.Value));
                        storage_form.RetrievingOrder(report);
                        storage_form.LoadData();
                        this.Close();
                }
        }
}
```