

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ
ОБЛІКУ ВИРОБНИЦТВА ДРОНІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE DRONE
PRODUCTION ACCOUNTING SYSTEM

Виконав: студент 4 курсу групи КНТ-110

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ВИНОГРАДОВ М.М.

(ПРІЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н., проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ВИНОГРАДОВА Миколи Миколайовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація системи обліку виробництва дронів.

Research and Software Implementation of the Drone Production Accounting System

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Прийняв виконане завдання
Основна частина	ГОФМАН Є. О., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Розробка архітектури програми.	2 тиждень	Розділ 2
4	Розробка програми.	3-4 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)


(підпис)

Микола ВІНОГРАДОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Підпис
(підпис)

Євгеній ГОФМАН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 124 с., 21 табл., 62 рис., 3 дод., 20 джерел.

СУБД, ЗАМОВЛЕННЯ, ДРОН, БАЗА ДАНИХ, КЛІЄНТ, СЕРВЕР, КОРИСТУВАЧ.

Об'єкт дослідження – програмне забезпечення системи обліку виробництва дронів.

Предмет дослідження – технології програмного забезпечення системи обліку виробництва дронів, які б забезпечили належну ефективність

Мета роботи – розробка системи, що забезпечує облік діяльності з виробництва дронів.

Матеріали, методи та технічні засоби: СКБД PostgreSQL, інтегроване середовище розробки Microsoft Visual Studio 2017 Community Edition, персональний комп'ютер з процесором Intel Pentium CPU N4200, оперативною пам'яттю 4 Гб, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи Microsoft Windows 10.

Результати. Розроблено програмну систему, яка призначена для обліку виробництва дронів.

Висновки. Розроблена система дозволяє спростити та підвищити надійність обліку виробництва дронів.

Галузь використання – підприємство, що займається виготовленням дронів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 124 pages, 21 tables, 62 figures, 3 appendixes, 20 sources.

DBMS, ORDER, DRONE, DATABASE, CLIENT, SERVER, USER.

Object of study is the software of the drone production accounting system.

Subject of study are software technologies for the accounting system of drone production, which would ensure proper efficiency.

The purpose of the work is to develop a system that provides accounting for drone production activities.

Materials, methods and tools: DBMS PostgreSQL, integrated development environments Microsoft Visual Studio 2017 Community Edition, personal computer with an Intel Pentium CPU N4200 processor, a 4GB RAM, a keyboard and a mouse, that runs under the control of the Microsoft Windows 10 operating system.

Results. A software system designed to account for drone production has been developed.

Conclusions. The developed system allows to simplify and increase reliability of drone production accounting.

The field of use is an enterprise engaged in the production of drones.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Короткий опис досліджуваної предметної області	10
1.2 Огляд існуючих рішень проблеми	10
1.2.1 Сервіс HugeProfit.....	11
1.2.2 Програмний продукт УкрСклад	11
1.2.3 Програмний продукт ТОРГСОФТ	12
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	13
1.3 Постановка завдання роботи.....	15
1.4 Висновки за розділом 1	15
2 Розробка архітектури програми.....	17
2.1 Опис основних вимог до програми	17
2.2 Структура програмної системи	17
2.3 Функціонування системи	19
2.4 Вимоги до окремих елементів структури.....	19
2.5 Вибір мови програмування та інструментарію для розробки програми	21
2.5.1 Мова програмування.....	21
2.5.2 Середовище розробки.....	22
2.5.3 База даних	23
2.6 Вимоги до технічних і програмних засобів.....	24
2.7 Проєктування бази даних	24
2.8 Висновки за розділом 2	29
3 Розробка програми	30
3.1 Алгоритм функціонування програми	30
3.2 Проєктування системи.....	32
3.3 Опис модулів системи	35
3.4 Проєктування бази даних	46
3.5 Висновки за розділом 3	52

4	Експлуатація, тестування та експериментальне дослідження програми	53
4.1	Опис застосування програми	53
4.2	Інструкція по роботі з програмою	53
4.2.1	Встановлення програми	53
4.2.2	Запуск програми	53
4.2.3	Методика роботи з програмою	53
4.3	Приклад роботи користувача з програмою	54
4.4	Тестування роботи програми	61
	Висновки	76
	Перелік джерел посилань	77
	Додаток А Технічне завдання	80
	Додаток Б Текст програми	86
	Додаток В Слайди презентації	119

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- IDE – Integrated Development Environment;
- SQL – Structured Query Language;
- БД – база даних;
- ОС – операційна система;
- ПЗ – програмне забезпечення;
- ПК – персональний комп'ютер;
- СКБД – система керування базами даних.

ВСТУП

У сучасному технологічному світі безпілотні літальні апарати, або дрони, займають важливе місце в різноманітних сферах, від військових дій до цивільного застосування. Швидкий темп розвитку цієї галузі спонукає до постійного вдосконалення технологій виробництва дронів, а також систем їх обліку та контролю. Виробництво та управління процесами їх створення стають важливими аспектами технологічного прогресу. У зв'язку з цим виникає потреба в ефективних системах обліку виробництва, які забезпечать не лише контроль якості продукції, але й оптимізацію процесів виготовлення, використання ресурсів та зменшення витрат.

Актуальність роботи полягає у тому, що програмне забезпечення такого характеру є завжди необхідним, оскільки дозволяє підвищити ефективність роботи з даними, та переглядати дані про стан виробництва у більш зручному форматі ніж у паперовому варіанті даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Короткий опис досліджуваної предметної області

Програмне забезпечення системи обліку виробництва дронів є інструментом, який допомагає в управлінні та контролі виробництва дронів підприємством.

Це особливо важливо для такого роду підприємств, оскільки правильний облік виробництва є основою для ефективного управління запасами, зменшення витрат грошей та покращення роботи з клієнтами.

Програмне забезпечення системи обліку виробництва дронів зазвичай має такі функції:

- облік запасів: система дозволяє вести детальний облік всіх товарів на складі, включаючи інформацію про кількість, місцезнаходження, стан, дату отримання та відправлення товару;
- замовлення та поставки: здійснення замовлення товарів від постачальників і відстеження процесу поставок;
- контроль за складськими рухами: фіксація всіх змін на складі, такі як отримання товару, переміщення між різними складськими зонами, продажі і повернення товару;
- зведена звітність: можливість отримати різноманітні звіти про стан запасів, оборотність, вартість товарів на складі та іншу аналітичну інформацію.

1.2 Огляд існуючих рішень проблеми

Розглянемо існуючі аналоги, які є найбільш поширеними та використовуються на даний момент для обліку виробництва, та приведемо їх переваги та недоліки за вищезазначеними критеріями.

1.2.1 Сервіс HugeProfit

HugeProfit — це CRM-система для обліку залишків товарів, контролю продажів та витрат, дозволяє створювати і відстежувати ТТН, працювати з клієнтською базою та наочно відображати доходи та витрати товарного бізнесу користувача (рис. 1.1) [1].

Основні переваги сервісу:

- простий облік невеликих виробництв;
- інтеграція з ПРРО;
- гнучка система створення складів;
- можливість друкувати цінники та чеки.

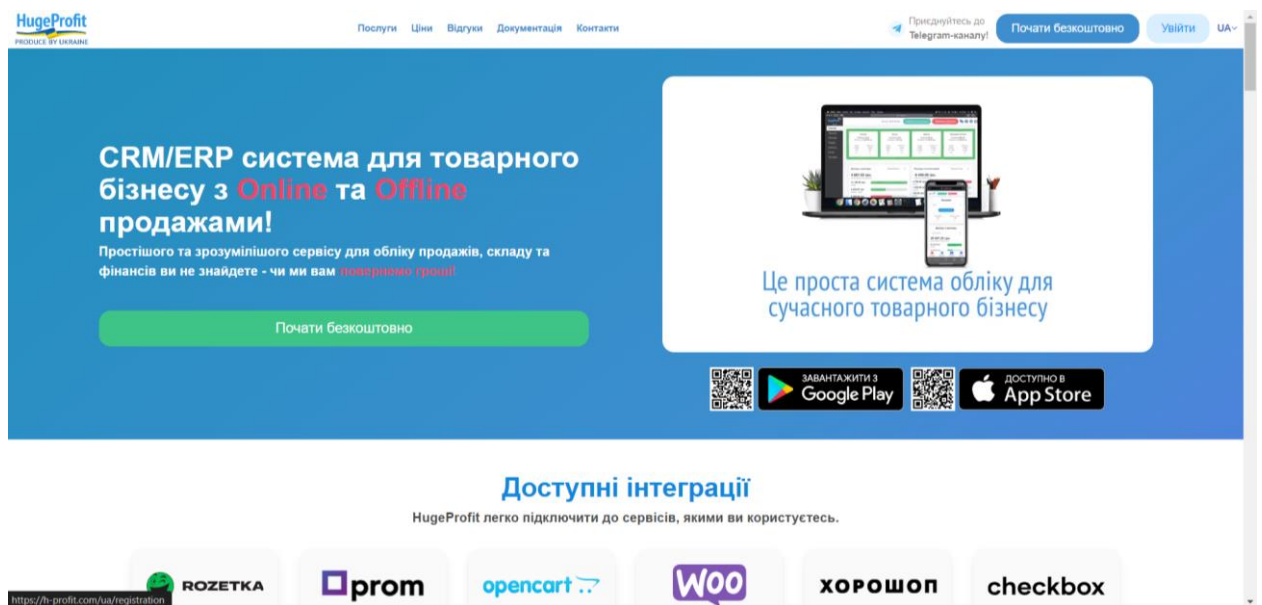


Рисунок 1.1 – Сервіс HugeProfit [1]

1.2.2 Програмний продукт УкрСклад

УкрСклад — функціональна програма, призначена для ведення обліку товарів на складах або фірмах (рис. 1.2) [2].

Основні переваги програмного продукту:

- ведення архіву будь-якої документації;

- можливість створювати довідники товарів, послуг, валют, клієнтів та активних фірм;
- підтримує роботу одразу багатьох користувачів роботи в локальній мережі з функцією спеціального збереження і в подальшому перенесення бланків на інші комп'ютери, де не встановлена програма.

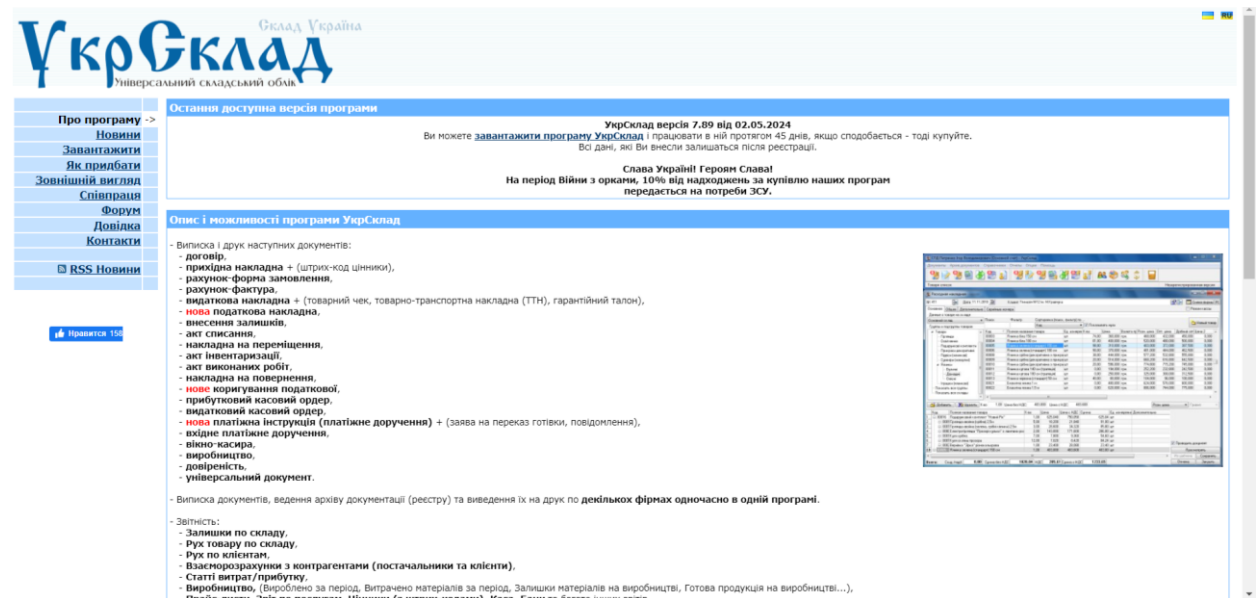


Рисунок 1.2 – Програмний продукт УкрСклад [2]

1.2.3 Програмний продукт ТОРГСОФТ

ТОРГСОФТ — це програма для автоматизації роздрібної та оптової торгівлі, інтернет-магазинів, складів та виробництв (рис. 1.3) [3].

Основні переваги програмного продукту:

- програма дозволяє робити повний цикл оприбуткування товару, формування замовлення і повернення товару постачальнику, контроль розрахунків з покупцями;
- виконання роботи з цінниками за штрихкодами, облік та уцінка, налаштування знижок.

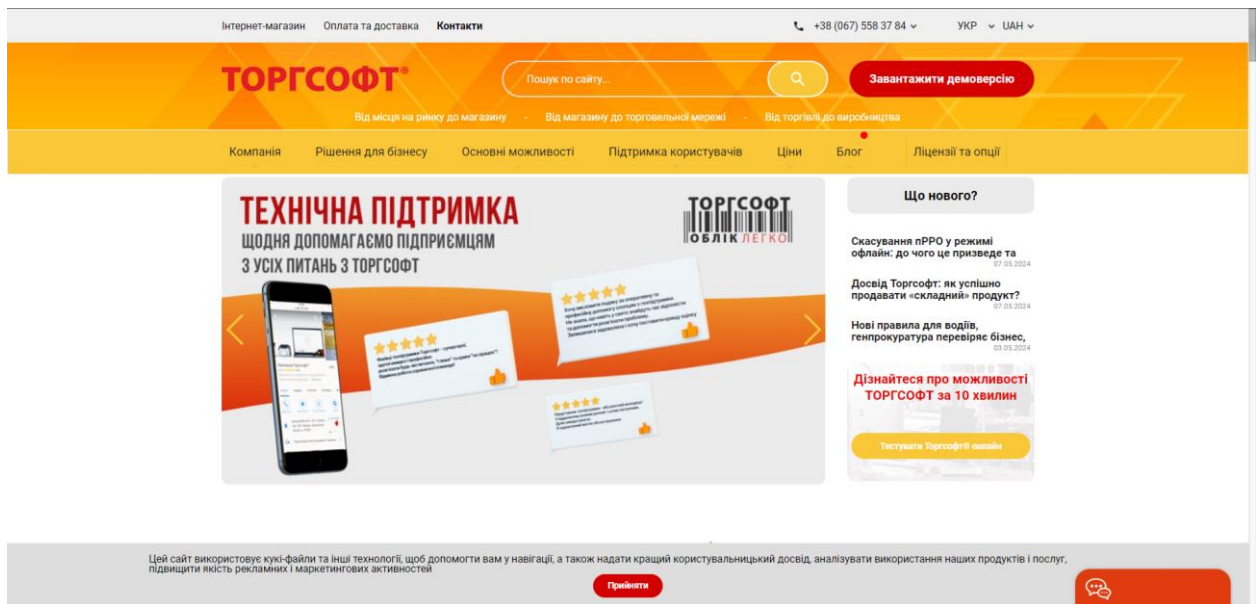


Рисунок 1.3 – Програмний продукт ТОРГСОФТ [3]

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день сервіси та програми, які дозволяють вести облік складу малого підприємства.

Але крім основних переваг, усі вони мають і недоліки. Наприклад:

- ТОРГСОФТ та HugeProfit в першу чергу розраховані на ведення комерційної діяльності, тому робота зі складом в них містить тільки базовий функціонал та не забезпечує формування звітів з цього роду даних;
- УкраСклад та ТОРГСОФТ не підтримують ролі користувачів, тому вони всі мають однаковий рівень доступу до даних;
- усі перелічені програмні продукти не є безкоштовними та мають дуже малий пробний період використання;
- УкрСклад має вкрай незручний інтерфейс, який дає користувачу одночасно дуже багато інформації, навіть якщо він її не запрошував і вона йому не потрібна.

Також дійсно існують переваги характерні для усіх програм, а саме можливість створювати власні звіти з таблиць та інструментарій для базової роботи з таблицями напряду.

У ході виконання дипломної роботи буде реалізовано програмний продукт, що буде мати наступні переваги, на відміну від існуючих аналогів:

- підтримка ролей користувачів, для контролювання доступу до даних;
- зручний інтерфейс, який дозволить отримувати користувачу тільки ту інформацію, яку він запросив;
- автоматизація формування звітів з таблиць;
- безкоштовність;
- гнучка система створення запитів до сервера;
- можливість роботи через локальну мережу, а не тільки через інтернет.

Проведено порівняння вже існуючих програмних продуктів та програми, яка буде розроблена в ході виконання роботи. Результат аналізу подано в табл. 1.1.

Таблиця 1.1 — Порівняння програмних продуктів

Функціонал	HugeProfit	Торгсофт	УкрСклад	ПП,що розробляється
Можливість роботи через Internet	+	+	-	+
Можливість роботи через локальну мережу	-	+	+	+
Зручний інтерфейс	+	+	-	+
Формування звітів	-	-	+	+
Підтримка різних типів користувачів	+	-	-	+

1.3 Постановка завдання роботи

Мета – Основною метою роботи є розробка системи обліку виробництва дронів.

Програмний продукт повинен забезпечувати виконання нижче перерахованих задач:

- відправка запиту від клієнта до сервера;
- отримання запиту на сервері;
- функції взаємодії із базою даних для клієнта і сервера (клієнт – формування запитів, для сервера – передача запитів до бази даних та перенаправлення результату клієнту);
- формування звітів на даних заснованих на отриманих з БД;
- виведення даних, отриманих від сервера на графічні форми користувача;
- авторизація та реєстрація користувачів;
- надання користувачу функціоналу в залежності від його ролі в системі;
- наявність валідації полів введення даних;
- чітке та зрозуміле відображення інформації;
- наявність головного меню для взаємодії з програмною системою;
- наявність графічних форм для різних типів користувачів.

1.4 Висновки за розділом 1

В першому розділі було розглянуто предметну область «Програмне забезпечення системи обліку виробництва дронів».

У ході аналізу було виявлено, що існуючі аналоги хоча і мають певні переваги у своїй сфері використання, але не здатні поєднати весь необхідний функціонал для забезпечення повної роботи зі складським обліком малого підприємства.

Актуальність була доведена тим, що розглянуті аналоги не містять повний функціонал необхідний для якісної роботи з обліку виробництва, що й стає причиною того, що програмний продукт, який розробляється є необхідністю, оскільки поєднає в собі весь потрібний функціонал, забезпечивши при цьому комфортну та гнучку роботу в області обліку виробництва дронів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Опис основних вимог до програми

Програма повинна:

- надавати можливість формування звітів за даними;
- мати клієнт-серверну архітектуру – система повинна складатися з клієнтського та серверного застосунку, між якими виконується пряму з'єднання [4];
- взаємодіяти з базою даних;
- містити захист від несанкціонованого доступу до даних в базі даних;
- містити графічні форми для продажу дронів, формування звітів, виготовлення дронів та їх компонентів, постачання матеріалів;
- містити графічну форму головного меню для взаємодії з клієнтською частиною системи.

2.2 Структура програмної системи

Програма складатиметься з двох типів модулів: клієнтські та серверні.

Клієнтські:

- графічна форма авторизації/реєстрації;
- графічна форма головного меню;
- модуль зв'язку (підключення до сервера, відправка даних на сервер та отримання відповіді);
- графічна форма продажі дронів;
- графічна форма виготовлення дронів;
- графічна форма постачання;
- графічна Форма формування звітів.

Серверні:

- модуль зв'язку (приймання запиту від клієнта та відправка відповіді);
- обробка отриманих запитів та формування запиту до бази даних;
- робота з базою даних (відправка сформованого запиту до бази даних та обробка результату);
- модуль безпеки даних (контроль доступу до даних).

Абстрактну модель структури програмної системи зображено на рис.

2.1.

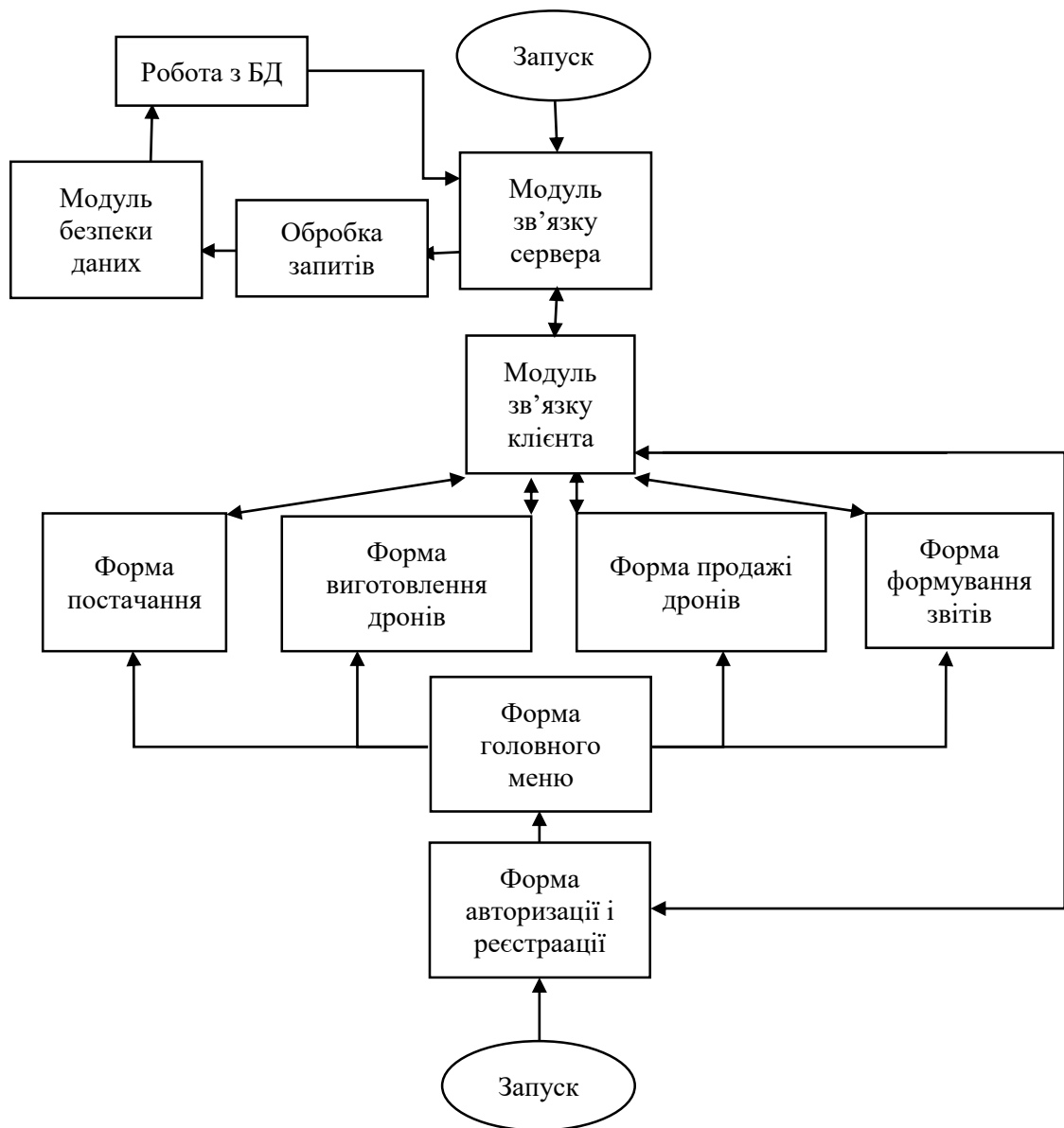


Рисунок 2.1 – Абстрактна модель структури програмної системи

2.3 Функціонування системи

Функціонування системи реалізується наступним чином: спочатку на сервері потрібно запустити базу даних PostgreSQL та серверну частину розроблювальної системи. Після запуску серверної частини – потрібно запустити клієнтську частину системи, на якій в подальшому буде забезпечуватися зв'язок з сервером.

2.4 Вимоги до окремих елементів структури

Вимоги до клієнтської частини:

Клієнтська частина системи повинна складатися з наступних модулів та компонентів:

а) графічна форма авторизації/реєстрації:

- 1) поля вводу даних;
- 2) кнопки здійснення авторизації та реєстрації;
- 3) графічна форма головного меню;
- 4) кнопки переходу на інші графічні форми;
- 5) рядок з інформацією про поточного користувача;

б) модуль зв'язку (підключення до сервера, відправка даних на сервер та отримання відповіді):

- 1) виконання з'єднання з сервером;
- 2) передача даних на сервер;
- 3) отримання даних від сервера;

в) графічна форма продажі дронів:

- 1) поля для вводу даних;
- 2) таблиця потенціальних покупців;
- 3) таблиця виготовлених дронів;
- 4) кнопки для здійснення продажі;

г) графічна форма виготовлення дронів:

- 1) поля для введення даних;
- 2) таблиця матеріалів;
- 3) таблиця компонентів;
- 4) кнопки для додання нових компонентів;
- 5) кнопки для додання нового дрона з вказаними компонентами;

д) графічна форма постачання:

- 1) таблиця постачальників з вказанням матеріал та ціни;
- 2) кнопки сортування таблиці постачальників;
- 3) поля для введення даних;
- 4) кнопки для здійснення замовлення постачальнику;

е) графічна форма формування звітів:

- 1) формування звіту з продаж;
- 2) формування звіту з постачання;
- 3) формування звіту з виготовлення дронів;
- 4) формування звіту з сумою грошей від продажу дронів.

Вимоги до серверної частини:

а) модуль зв'язку (приймання запиту від клієнта та відправка відповіді):

- 1) виконання з'єднання з клієнтом;
- 2) передача даних клієнту;
- 3) отримання даних від клієнта;

б) обробка отриманих запитів та формування запиту до бази даних:

- 1) формування запиту до бази даних;
- 2) обробка запиту та передача його у модуль безпеки даних;

в) робота з базою даних (відправка сформованого запиту до бази даних та обробка результату):

- 1) відправка сформованого запиту у базу даних;
- 2) отримання результату від бази даних;
- 3) формування відповіді для відправки клієнту;

г) модуль безпеки даних (контроль доступу до даних):

- 1) перевірка запиту на зайві елементи;
- 2) відправка запиту на формування.

2.5 Вибір мови програмування та інструментарію для розробки програми

Для розробки програмного забезпечення була використана мова програмування C# з .NET Framework та середовище розробки Microsoft Visual Studio 2017 Community Edition. Для зберігання даних у базі даних використовувалася СКБД PostgreSQL.

2.5.1 Мова програмування

C# є сучасною, об'єктно-орієнтованою мовою програмування. Вона є частиною платформи .NET і широко використовується для розробки різних типів додатків, включаючи вебдодатки, настільні програми, мобільні додатки та ігри [5].

Синтаксис C# схожий на синтаксис інших мов програмування, таких як C++ та Java. Вона базується на блоках коду, використовує фігурні дужки для визначення областей видимості і має сувору типізацію [6].

C# є сильно типізованою мовою, що означає, що типи змінних визначаються на етапі компіляції. Це дозволяє виявляти помилки на етапі розробки і забезпечує більшу безпеку та надійність програми [7].

Многопоточковість: C# надає підтримку многопоточковості та асинхронного програмування, що дозволяє створювати ефективні та продуктивні додатки, які можуть виконувати кілька завдань паралельно [8].

Обрана мова програмування використовує автоматичне керування пам'яттю за допомогою збірки сміття. Це значно спрощує процес управління та допомагає уникнути витоків пам'яті [9].

Має великий арсенал вбудованих бібліотек, які дозволяють працювати з базами даних, оброблювати виняткові ситуації, а також створювати клієнт-серверні застосунки [10].

Порівняльну таблицю функціональності наведено в табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

Функціонал	C++	C#	Python	Java
Підтримка .NET	+	+	-	-
Многопотоковість	+	+	+-	+
Автоматичне керування пам'яттю	+	-	+	+
Робота з БД	+	+	+-	+
Типізація	+	+	+-	+

2.5.2 Середовище розробки

Microsoft Visual Studio 2017 Community Edition є безкоштовним інтегрованим середовищем розробки від Microsoft для розробки настільних та мобільних додатків, вебдодатків, вебсервісів, хмарних сервісів тощо [11]. Є основим для розробки за допомогою платформи .NET Framework [12]. Її Редактор коду має безліч функцій, які допомагають підвищити продуктивність роботи та спростити процес написання коду програми: автодоповнення, перевірка синтаксису, інтегрована підтримка налагодження і багато іншого [13]. Крім цього, надає потужні інструменти для налагодження програм: можна встановлювати точки зупинки, відстежувати значення змінних, виконувати кроки налагодження та аналізувати потік виконання програми, щоб виявити та виправити помилки [14].

Microsoft Visual Studio 2017 Community Edition підтримує розширення, які дозволяють додавати додаткову функціональність та інтегрувати сторонні інструменти [15].

Середовище розробки підтримується лише на операційних системах Windows, хоча і є способи запуску на інших системах, але такі версії мають значно обмежений функціонал у порівнянні зі звичайною [16].

Обране середовище розробки підтримує роботу із репозиторіями Git і Team Foundation Server [17].

Має безкоштовну версію Community Edition, яка не має функціональних обмежень [18].

В табл. 2.2 наведено порівняльну характеристик середовищ розробки.

Таблиця 2.2 – Порівняння середовищ розробки

Характеристика	Microsoft Visual Studio 2017 Community Edition	JetBrains Rider	MonoDevelop
Багатоплаформеність	-	+	-
Швидкість модернізації	+	-	-
Наявність візуального конструктора	+	+	+
Підтримка репозиторіїв	+	+	+
Доступність	+	+	-
Підтримка .Net	+	+	+/-

2.5.3 База даних

PostgreSQL — це потужна об'єктно-реляційна база даних з відкритим вихідним кодом, активна розробка якої триває багато років, завдяки чому вона заслужила міцну репутацію надійності і продуктивності [19]. Вона підтримує

як структуровану мову запитів так і JSON Для реляційних і нереляційних типів баз даних [20].

Основні переваги PostgreSQL:

- безкоштовна та доступна для всіх;
- наявність підтримки асинхронної реплікації даних;
- присутність резервного копіювання;
- підтримка табличних просторів;
- підтримка всіх операційних систем;
- можливість визначати власні типи даних;
- введення журналу, що дозволяє відмовостійкість.

2.6 Вимоги до технічних і програмних засобів

Для роботи програми потрібно: встановлена та запущена на сервері база даних PostgreSQL, інтегроване середовище розробки Microsoft Visual Studio 2017 Community Edition (для відлагодження коду), персональний комп'ютер з процесором Intel Pentium CPU N4200, оперативною пам'яттю 4 Гб, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи Microsoft Windows 10

2.7 Проєктування бази даних

Концептуальну модель бази даних наведено на рис. 2.2.

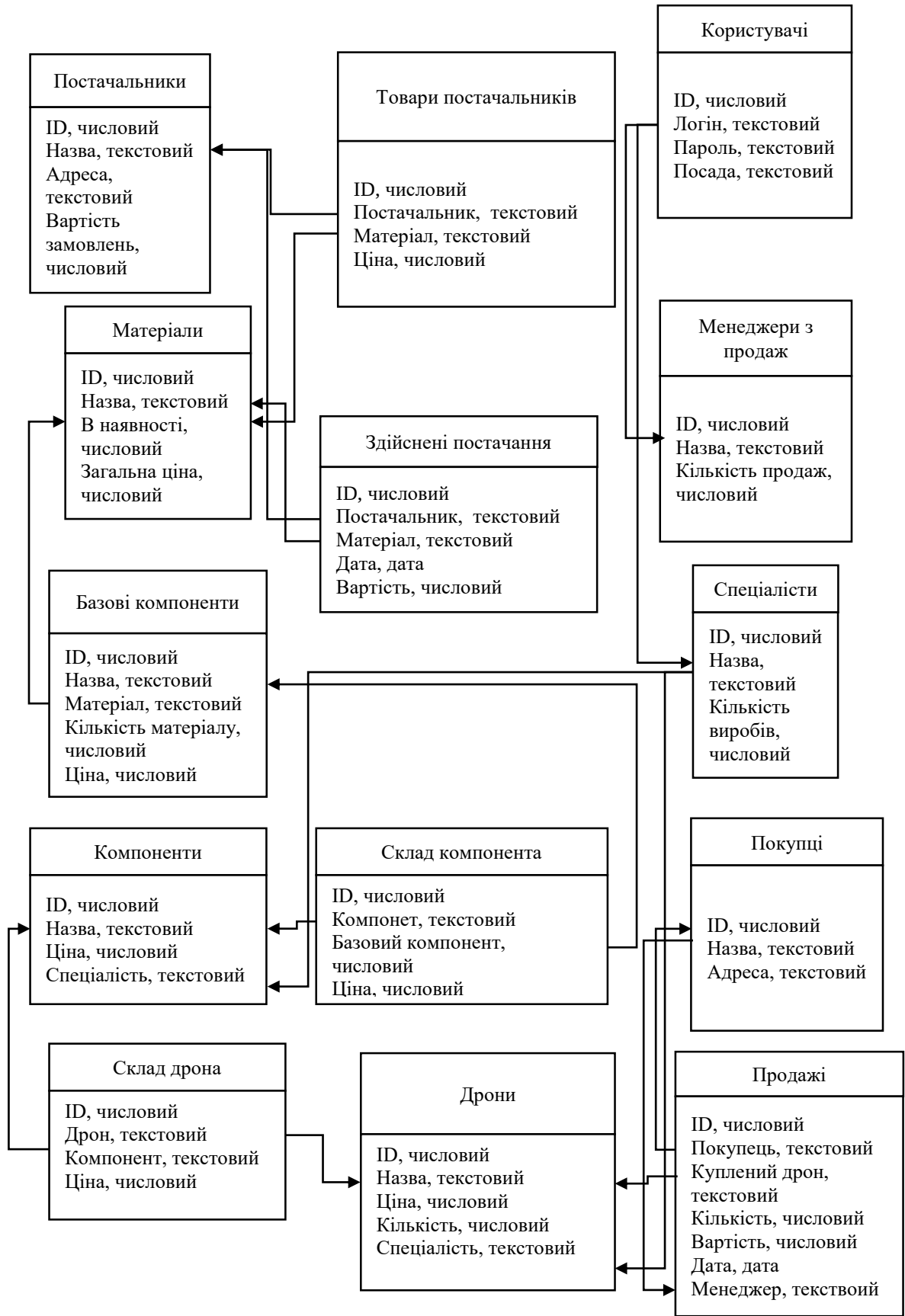


Рисунок 2.2 – Концептуальна модель бази даних

Опис сутностей моделі бази даних наведено в табл. 2.3.

Таблиця 2.3 – Опис концептуальної моделі бази даних

№	Сутність	Опис	Атрибут	Тип даних	Опис
1	2	3	4	5	6
1	Користувачі	Зберігає дані про користувачів системи	ID	Числовий	Айді користувача в таблиці
			Логін	Текстовий	Логін користувача
			Пароль	Текстовий	Пароль користувача
			Посада	Текстовий	Посада користувача
2	Постачальники	Представляє інформацію про постачальників	ID	Числовий	Айді постачальника в таблиці
			Назва	Текстовий	Назва постачальника
			Адреса	Текстовий	Адреса постачальника
			Вартість замовлень	Числовий	Вартість всіх здійснених замовлень у постачальника
3	Товари постачальників	Список товарів постачальників	ID	Числовий	Айді товару постачальника у таблиці
			Постачальник	Текстовий	Назва постачальника який здійснює постачання
			Матеріал	Текстовий	Матеріал який постачальник може постачати
			Ціна	Числовий	Ціна за одну тонну матеріалу
4	Здійснені постачання	Список здійснених постачань	ID	Числовий	Айді постачання в таблиці
			Постачальник	Текстовий	Постачальник який здійснив постачання
			Матеріал	Текстовий	Матеріал, що постачається
			Вартість	Числовий	Вартість постачання
			Дата	Дата	Дата здійснення постачання
5	Матеріали	Список матеріалів	ID	Числовий	Айді матеріалу в таблиці
			Назва	Текстовий	Назва матеріалу

Продовження таблиці 2.3

1	2	3	4	5	6
5	Матеріали	Список матеріалів	В наявності	Числовий	Вага наявного матеріалу в тонах
			Загальна ціна	Числовий	Загальна ціна всього наявного матеріалу
6	Базові компоненти	Список базових компонентів	ID	Числовий	Айді базового компонента
			Назва	Текстовий	Назва базового компоненту
			Матеріал	Текстовий	Назва використаного матеріалу
			Кількість матеріалу	Числовий	Кількість використаного матеріалу у грамах
			Ціна	Числовий	Ціна виоготовлення компонента
7	Склад компонента	Зберігає інформацію про склад компонента	ID	Числовий	Айді базового компонента з якого складається компонент
			Компонент	Текстовий	Назва компоненту
			Базовий компонент	Текстовий	Базовий компонент, який входить до складу поаного компоненту
			Ціна	Числовий	Ціна базового компоненту
8	Компоненти	Список компонентів для виготовлення дронів	ID	Числовий	Айді компоенту в таблиці
			Назва	Текстовий	Назва компоненту
			Ціна	Числовий	Ціна компонента
			Спеціалість	Текстовий	Спеціалість який займався виготовленням
9	Склад дрона	Список компонентів з яких складається дрон	ID	Числовий	Айді компонента, який входить до складу дрона
			Дрон	Текстовий	Назва дрона
			Компонент	Текстовий	Назва компоненту, який входить до складу дрона

Продовження таблиці 2.3

1	2	3	4	5	6
9	Склад дрона	Список компонентів з яких складається дрон	Ціна	Числовий	Ціна компонента дрона
10	Дрони	Список виготовляємих дронів	ID	Числовий	Айді дрону в таблиці
			Назва	Текстовий	Назва дрону
			Ціна	Числовий	Ціна одного дрону
			Кількість	Числовий	Кількість дронів
			Спеціаліст	Текстовий	Спеціаліст який займався виготовленням
11	Покупці	Покупці яким здійснюється продаж дронів	ID	Числовий	Айді покупця в таблиці
			Назва	Текстовий	Назва покупця
			Адреса	Текстовий	Адреса покупця
12	Продажі	Продажі дронів	ID	Числовий	Айді здійсненої продажі
			Покупець	Текстовий	Назва покупця
			Куплений дрон	Текстовий	Назва купленого дрона
			Кількість	Числовий	Кількість проданих дронів
			Вартість	Числовий	Вартість продажі
			Дата	Дата	Дата здійснення продажі
			Менеджер	Текстовий	Менеджер, що займався даною продажою
13	Менеджери з продаж	Користувачі, що займаються продажами	ID	Числовий	Айді менеджера з продаж
			Назва	Текстовий	Назва менеджера з продаж
			Кількість продаж	Числовий	Кількість виконаних продаж
14	Спеціалісти	Спеціалісти з виготовлення дронів	ID	Числовий	Айді спеціаліста
			Назва	Текстовий	Назва спеціаліста
			Кількість виробів	Числовий	Кількість виготовлених виробів спеціалістом

2.8 Висновки за розділом 2

Як результат роботи, у другому розділі було описано структуру програми, основні вимоги до програми та її окремих елементів структури. Обрано та обґрунтовано вибір мови програмування, бази даних та середовища розробки. Було описано функціонування системи та наведено вимоги до технічних і програмних засобів. Окрім цього була побудована та описана концептуальна модель бази даних проєкта.

- встановлення параметрів підключення – алгоритм, мета якого встановити параметри підключення клієнта до сервера (ір і порт). Параметри, які встановленні у цьому алгоритмі будуть використовуватися під час подальших підключень до сервера;
- авторизація – алгоритм, мета якого встановити параметри підключення клієнта до сервера (ір і порт). Параметри, які встановленні у цьому алгоритмі будуть використовуватися під час подальших підключень до сервера;
- форма навігації – виконує налаштування форми навігації;
- форма менеджера з продаж. Завантаження даних – виконує запити до сервера на отримання списків дронів на складі, покупців, виконаних продаж. Отримані дані заносить до виділених для них списків;
- форма менеджера з продаж. Запити до сервера – комплекс алгоритмів, які призначені для відправки на сервер запиту на додання або оновлення даних з таблиць, що відносяться замовлень. Дані для таких запитів беруться з текстових полів на формі;
- форма спеціаліста. Завантаження даних – виконує запити до сервера на отримання списків матеріалів, компонентів, компонентів у дронах, базових компонентів, склад компонентів. Отримані дані заносить до виділених для них списків;
- форма спеціаліста. Запити до сервера – комплекс алгоритмів, які призначені для відправки на сервер запиту на додання або оновлення даних з таблиць компонентів, матеріалів, компонентів у дронах, складу компонентів, базових компонентів та таблиці дронів. Дані для таких запитів беруться з текстових полів на формі;
- форма складського менеджера. Завантаження даних – виконує запити до сервера на отримання списків матеріалів, постачальник та їх товарів. Отримані дані заносить до виділених для них списків;
- форма складського менеджера. Запити до сервера – комплекс алгоритмів, які призначені для відправки на сервер запиту на додання або

оновлення даних з таблиць матеріалів, постачальників та їх товарів. Дані для таких запитів беруться з текстових полів на формі;

- відправка даних серверу – приймає на вході строку даних, перетворює її в масив байтів, передає серверу;
- отримання результату від сервера – отримує від сервера масив байтів, перетворює їх в строку;
- отримання даних від клієнта – прослуховує вхідні підключення, у випадку наявного, приймає масив байтів, перетворює його в строку, передає функції захисту даних, отриманий результат відправляє у функцію відправки даних клієнту;
- відправлення даних клієнту – приймає на вході строку, перетворює її в масив байтів та відправляє підключеному клієнту;
- перевірка на безпеку даних – перевіряє запит клієнта на наявність недоступних йому даних та можливостей;
- обробка і перенаправлення запиту до бази даних – оброблює запит та перенаправляє його у базу даних. Перенаправлення у потрібну функцію модуля роботи з БД відбувається за допомогою кодового слова;
- запити до БД – комплекс алгоритмів, які виконують запити до бази даних на вибірку, видалення оновлення, додавання даних. Усі ці алгоритми працюють по принципу: створення строки sql запиту відповідно до потреб, відкриття зв'язку з базою даних, відправлення запиту, отримання результату (у випадку запиту на вибірку даних), закриття з'єднання з базою даних.

3.2 Проєктування системи

Архітектура розробленої клієнтської частини програмної системи, продемонстровано на рис. 3.2.

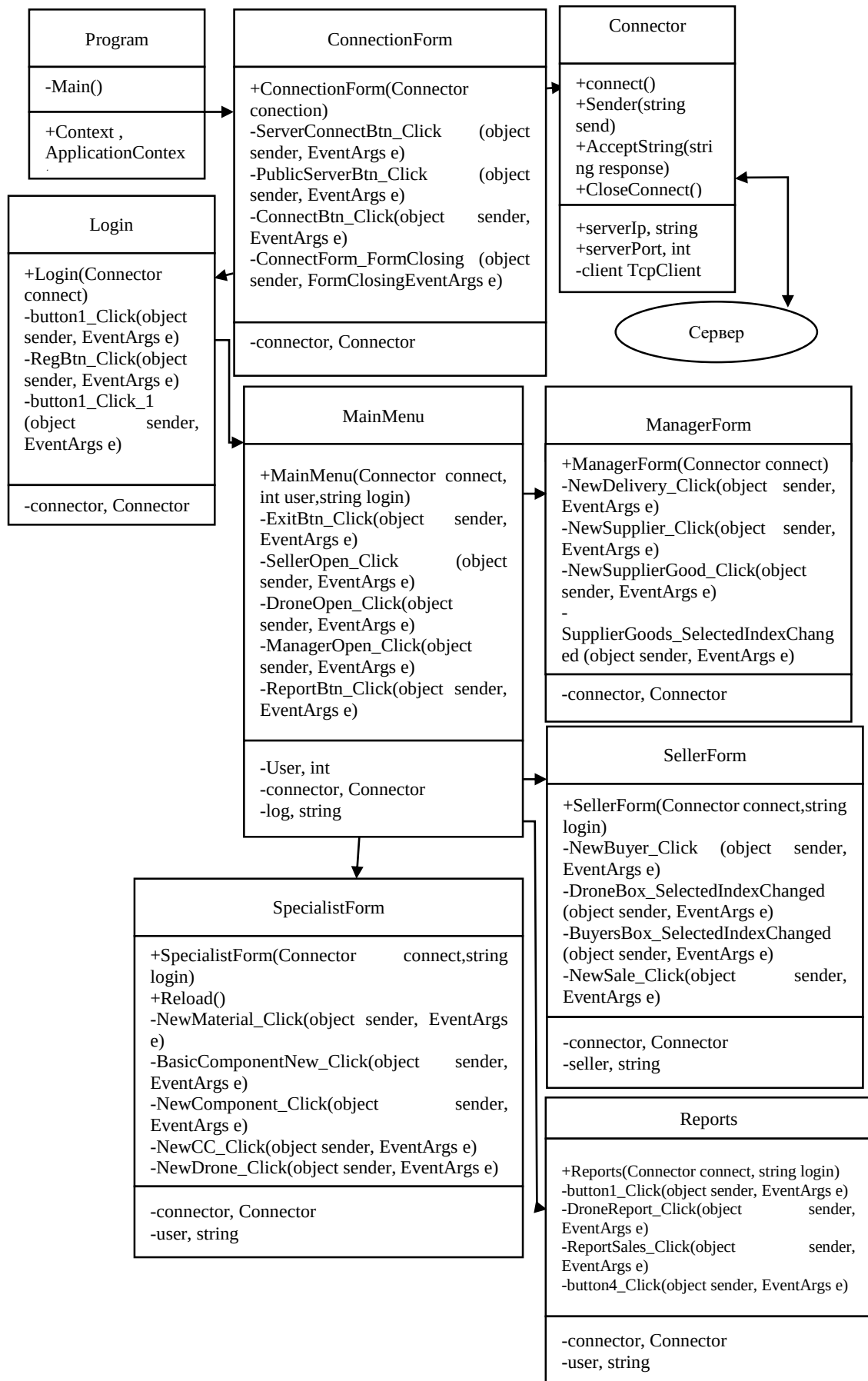


Рисунок 3.2 – Проектування та розроблення архітектури системи (Клієнт)

Архітектуру розробленої серверної частини програмної системи продемонстровано на рис 3.3.

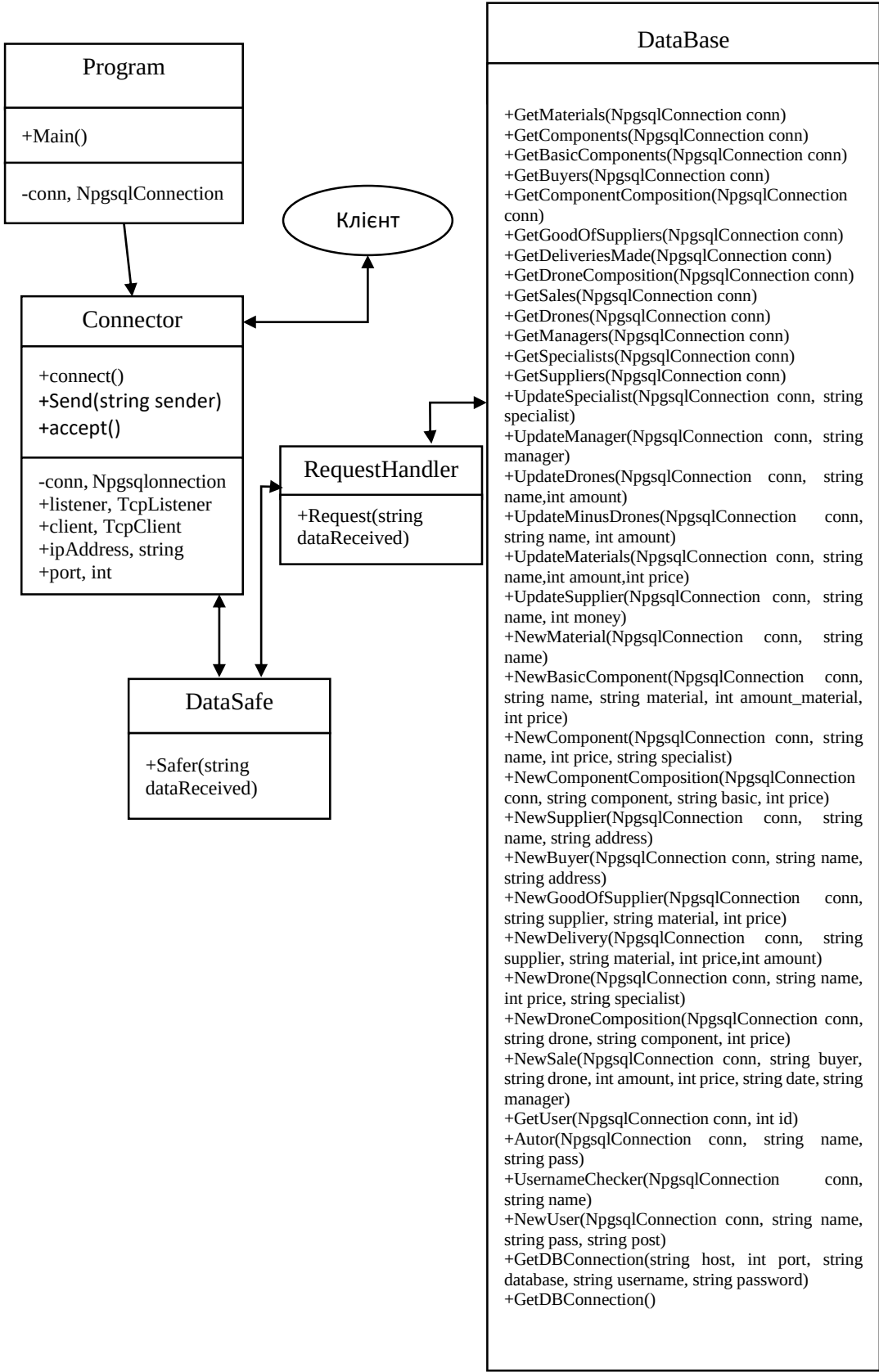


Рисунок 3.3 – Проектування та розроблення архітектури системи (Сервер)

3.3 Опис модулів системи

Опис модулів розробленої клієнтської частини системи наведено в табл. 3.1.

Таблиця 3.1 – Опис модулів системи (Клієнт)

Назва	Опис	Основні функції	Опис функції
1	2	3	4
Program	Модуль початку роботи	static void Main()	Функція входу в програму.
ConnectionForm	Графічна форма встановлення параметрів підключення. Початкова форма.	public ConnectionForm(Connector conection)	Виконує ініціалізацію компонентів форми.
		private void ServerConnectBtn_Click(object sender, EventArgs e)	Відкриває форму авторизації використовуючи адресу сервера за замовчуванням.
		private void PublicServerBtn_Click(object sender, EventArgs e)	Перемикає форму у режим встановлення параметрів підключення.
		private void ConnectBtn_Click(object sender, EventArgs e)	Встановлює параметри підключення. Відкриває форму авторизації.
		private void ConnectForm_FormClosing(object sender, FormClosingEventArgs e)	Закриває з'єднання з сервером при закритті форми.
Connector	Модуль зв'язку з сервером	public void connect()	Виконує підключення до сервера.
		public void Sender(string send)	Надсилає серверу дані.
		public string AcceptString(string response)	Отримує дані від сервера.
		public void CloseConnect()	Закриває з'єднання з сервером.
Login	Модуль графічної форми авторизації.	public Login(Connector connect)	Виконує ініціалізацію компонентів форми.

Продовження таблиці 3.1

1	2	3	4
Login	Модуль графічної форми авторизації.	private void button1_Click(object sender, EventArgs e)	Виконує авторизацію користувача в системі.
		private void RegBtn_Click(object sender, EventArgs e)	Виконує реєстрацію користувача в системі.
		private void button1_Click_1(object sender, EventArgs e)	Переводє форму у режим реєстрації.
MainMenu	Навігаційна форма програми.	public MainMenu(Connector connect, int user,string login)	Виконує ініціалізацію компонентів форми.
		private void ExitBtn_Click(object sender, EventArgs e)	Виконує закриття форми та вихід з програми.
		private void SellerOpen_Click(object sender, EventArgs e)	Відкриває графічну форму менеджера з продаж.
		private void DroneOpen_Click(object sender, EventArgs e)	Відкриває графічну форму спеціаліста з виготовлення дронів.
		private void ManagerOpen_Click(object sender, EventArgs e)	Відкриває графічну форму складського менеджера.
		private void ReportBtn_Click(object sender, EventArgs e)	Відкриває графічну форму створення звітів.
ManagerForm	Графічна форма складського менеджера.	public ManagerForm(Connector connect)	Виконує ініціалізацію компонентів форми, виконує запит до сервера на отримання даних для таблиць.
		private void NewDelivery_Click(object sender, EventArgs e)	Виконує додавання нового постачання.
		private void NewSupplier_Click(object sender, EventArgs e)	Виконує додавання нового постачальника.

Продовження таблиці 3.1

1	2	3	4
ManagerForm	Графічна форма складського менеджера.	private void NewSupplierGood_Click(object sender, EventArgs e)	Додання нового товару доступного для постачання.
		private void SupplierGoods_SelectedIndexChanged(object sender, EventArgs e)	При виборі елемента списку встановлює дані цього елемента у поля для введення даних.
SellerForm	Графічна форма менеджера з продаж.	public SellerForm(Connector connect,string login)	Виконує ініціалізацію компонентів форми, виконує запит до сервера на отримання даних для таблиць.
		private void NewBuyer_Click(object sender, EventArgs e)	Додає нового покупця у базу даних.
		private void DroneBox_SelectedIndexChanged(object sender, EventArgs e)	При виборі елемента списку встановлює дані цього елемента у поля для введення даних дрона.
		private void BuyersBox_SelectedIndexChanged(object sender, EventArgs e)	При виборі елемента списку встановлює дані цього елемента у поля для введення даних покупця.
		private void NewSale_Click(object sender, EventArgs e)	Додає нову продаж у базу даних.
SpecilistForm	Графічна форма спеціаліста з виготовлення дронів.	public SpecialistForm(Connector connect,string login)	Виконує ініціалізацію компонентів форми.
		public void Reload()	Виконує запити до сервера на отримання даних таблиць та встановлення їх для списків на формі.
		private void NewMaterial_Click(object sender, EventArgs e)	Додання нового матеріалу до бази даних.
		private void BasicComponentNew_Click(object sender, EventArgs e)	Додання нового базового компоненту у базу даних.

Продовження таблиці 3.1

1	2	3	4
SpecilistForm	Графічна форма спеціаліста з виготовлення дронів.	private void NewComponent_Click(object sender, EventArgs e)	Додання нового компонента у базу даних.
		private void NewCC_Click(object sender, EventArgs e)	Додання до складу компонента нового базового компонента.
		private void NewDrone_Click(object sender, EventArgs e)	Додання нового дрона у базу даних.
		private void AddDrone_Click(object sender, EventArgs e)	Додання одного дрона обраного типу до бази даних.
		private void NewDC_Click(object sender, EventArgs e)	Додання до складу дрона нового компоненту.
Reports	Графічна форма для створення звітів.	public Reports(Connector connect, string login)	Виконує ініціалізацію компонентів форми.
		private void button1_Click(object sender, EventArgs e)	Формує звіт з поставчань.
		private void DroneReport_Click(object sender, EventArgs e)	Формує звіт з дронів на складі.
		private void ReportSales_Click(object sender, EventArgs e)	Формує звіт з виконаних продаж дронів.
		private void button4_Click(object sender, EventArgs e)	Розраховує сумарну вартість продаж дронів.

Графічна форма встановлення параметрів підключення продемонстрована на рис. 3.4.

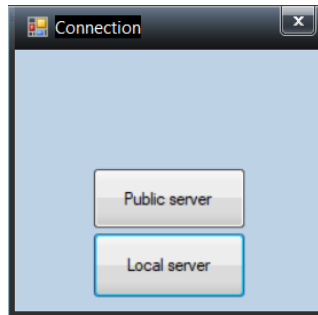


Рисунок 3.4 – Графічна форма встановлення параметрів підключення

Графічну форму авторизації представлено на рис. 3.5.

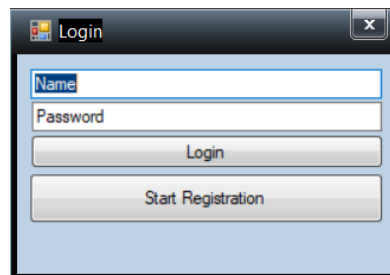


Рисунок 3.5 – Форма авторизації

Графічну навігаційну форму представлено на рис. 3.6.

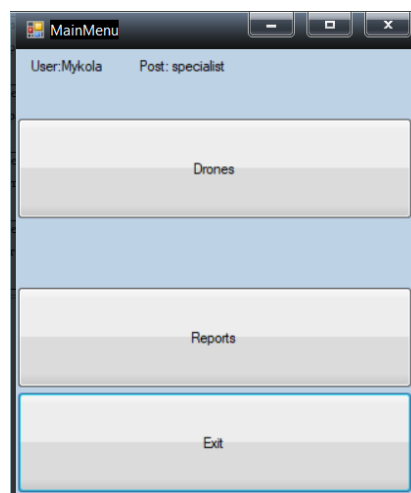


Рисунок 3.6 – Навігаційна форма

Графічну форму складського менеджера продемонстровано на рис. 3.7.

Рисунок 3.7 – Графічна форма складського менеджера

Графічну форму менеджера з продаж представлено на рис. 3.8.

Рисунок 3.8 – Графічна форма менеджера з продаж

Графічну форму спеціаліста з виготовлення дронів представлено на рис. 3.9.

The 'SpecialistForm' window is divided into six main sections:

- Top Left:** Lists existing components with fields for Name, Price, and Specialist.
- Top Middle:** Lists existing basic components with fields for Name, Material, Amount of material, and Price.
- Top Right:** Lists existing materials with fields for Material, In storage, and Total price.
- Middle Left:** Form for adding a new component with fields for Name, Price, and an 'Add component' button.
- Middle Middle:** Form for adding a new basic component with fields for Name, Material, Amount of material, Price, and an 'Add basic component' button.
- Middle Right:** Form for adding a new material with a 'Material' field and an 'Add material' button.
- Bottom Left:** Form for adding a new component composition with fields for Component, BasicComponent, and Price, and an 'Add basic component' button.
- Bottom Middle:** Form for adding a new drone with fields for Name, Price, and an 'Add drone' button.
- Bottom Right:** Form for adding a new drone composition with fields for Drone, Component, and Price, and an 'Add component into drone' button.

Рисунок 3.9 – Графічна форма спеціаліста з виготовлення дронів

Графічну форму для створення звітів продемонстровано на рис. 3.10.

The 'Reports' window displays a list of transaction records with the following fields: Id, Buyer, Drone, First, Amount, Price, Date, and Manager. Below the list are four buttons for generating reports:

- Report deliveries
- Report drones
- Calculate money from drone selling
- Report sales

Рисунок 3.10 – Графічна форма для створення звітів

Опис модулів розробленої серверної програмної частини системи наведено в табл. 3.2.

Таблиця 3.2 – Опис модулів системи (Сервер)

Назва	Опис	Основні функції	Опис функції
1	2	3	4
Program	Модуль початку роботи сервера	static void Main()	Функція входу в програму. Запускає сервер та перевіряє доступ до бази даних.
Connector	Модуль підключення	public void connect()	Починає прослуховування вхідних підключень.
		public void accept()	При отриманні даних від клієнта перенаправляє їх до модулю захисту даних. Після отриманні результату запиту викликає функцію надсилання відповіді.
		public void Send(string sender)	Надсилає відповідь користувача.
DataSafe	Модуль безпеки даних	public string Safer(string dataReceived)	Перевіряє запит клієнта на присутність в ньому не бажаних елементів, що надають можливість впливати на базу даних. Далі надсилає його на обробку.
RequestHandler	Модуль обробки та перенаправлення даних (реалізацію модулю наведено в додатку Б)	public string Request(string dataReceived)	Виконує обробку та перенаправлення запиту клієнта до функцій модулю бази даних.
DataBase	Модуль роботи з базою даних	public string[] GetMaterials(NpgsqlConnection conn) (додаток Б.3.1)	Виконує запит до бази даних на отримання даних з таблиці матеріалів.
		public string[] GetComponents(NpgsqlConnection conn) (додаток Б.3.2)	Виконує запит до бази даних на отримання даних з таблиці компонентів.
		public string[] GetBasicComponents(NpgsqlConnection conn) (додаток Б.3.3)	Виконує запит до бази даних на отримання даних з таблиці базових компонентів.

Продовження таблиці 3.2

1	2	3	4
DataBase	Модуль роботи з базою даних	public string[] GetBuyers(NpgsqlConnection conn) (додаток Б.3.4)	Виконує запит до бази даних на отримання даних з таблиці покупців.
		public string[] GetComponentComposition(NpgsqlConnection conn) (додаток Б.3.5)	Виконує запит до бази даних на отримання даних з таблиці складу компонентів.
		public string[] GetGoodOfSuppliers(NpgsqlConnection conn) (додаток Б.3.6)	Виконує запит до бази даних на отримання даних з таблиці товарів постачальників.
		public string[] GetDeliveriesMade(NpgsqlConnection conn) (додаток Б.3.7)	Виконує запит до бази даних на отримання даних з таблиці зроблених поставчань.
		public string[] GetDroneComposition(NpgsqlConnection conn) (додаток Б.3.8)	Виконує запит до бази даних на отримання даних з таблиці складу дронів.
		public string[] GetSales(NpgsqlConnection conn) (додаток Б.3.9)	Виконує запит до бази даних на отримання даних з таблиці продаж.
		public string[] GetDrones(NpgsqlConnection conn) (додаток Б.3.10)	Виконує запит до бази даних на отримання даних з таблиці дронів.
		public string[] GetManagers(NpgsqlConnection conn) (додаток Б.3.11)	Виконує запит до бази даних на отримання даних з таблиці менеджерів з продаж.
		public string[] GetSpecialists(NpgsqlConnection conn) (додаток Б.3.12)	Виконує запит до бази даних на отримання даних з таблиці спеціалістів з виготовлення дронів.
		public string[] GetSuppliers(NpgsqlConnection conn) (додаток Б.3.13)	Виконує запит до бази даних на отримання даних з таблиці постачальників.

Продовження таблиці 3.2

1	2	3	4
DataBase	Модуль роботи з базою даних	public void UpdateSpecialist(NpgsqlConnection conn, string specialist) (додаток Б.3.14)	Виконує запит до бази даних на оновлення даних таблиці спеціалістів з виготовлення дронів. Додає до кількості виготовлених виробів одиницю.
		public void UpdateManager(NpgsqlConnection conn, string manager) (додаток Б.3.15)	Виконує запит до бази даних на оновлення даних таблиці менеджерів з продаж. Додає до кількості здійснених продаж одиницю.
		public void UpdateDrones(NpgsqlConnection conn, string name, int amount) (додаток Б.3.16)	Виконує запит до бази даних на оновлення даних таблиці дронів. Додає задану кількість дронів.
		public void UpdateMinusDrones(NpgsqlConnection conn, string name, int amount) (додаток Б.3.17)	Виконує запит до бази даних на оновлення даних таблиці дронів. Віднімає задану кількість дронів з складу.
		public void UpdateMaterials(NpgsqlConnection conn, string name, int amount, int price) (додаток Б.3.18)	Виконує запит до бази даних на оновлення даних таблиці матеріалів. Додає задану кількість матеріалу та його вартість на склад.
		public void UpdateSupplier(NpgsqlConnection conn, string name, int money) (додаток Б.3.19)	Виконує запит до бази даних на оновлення даних таблиці постачальників. Додає до загальної вартості всіх поставок задану суму.
		public string[] NewMaterial(NpgsqlConnection conn, string name) (додаток Б.3.20)	Виконує запит до бази даних на додавання нового запису у таблицю матеріалів.

Продовження таблиці 3.2

1	2	3	4
DataBase	Модуль роботи з базою даних	public string[] NewBasicComponent(NpgsqlCo nnection conn, string name, string material, int amount_material, int price) (додаток Б.3.21)	Виконує запит до бази даних на додання нового запису у таблицю базових компонентів.
		public string[] NewComponent(NpgsqlConnect ion conn, string name, int price, string specialist) (додаток Б.3.22)	Виконує запит до бази даних на додання нового запису у таблицю компонентів.
		public string[] NewComponentComposition(Np gsqlConnection conn, string component, string basic, int price) (додаток Б.3.23)	Виконує запит до бази даних на додання нового запису у таблицю складу компонентів.
		public string[] NewSupplier(NpgsqlConnection conn, string name, string address) (додаток Б.3.24)	Виконує запит до бази даних на додання нового запису у таблицю постачальників.
		public string[] NewBuyer(NpgsqlConnection conn, string name, string address) (додаток Б.3.25)	Виконує запит до бази даних на додання нового запису у таблицю покупців.
		public string[] NewGoodOfSupplier(NpgsqlCo nnection conn, string supplier, string material, int price) (додаток Б.3.26)	Виконує запит до бази даних на додання нового запису у таблицю товарів постачальників.
		public string[] NewDelivery(NpgsqlConnection conn, string supplier, string material, int price,int amount) (додаток Б.3.27)	Виконує запит до бази даних на додання нового запису у таблицю зроблених поставок
		public string[] NewDrone(NpgsqlConnection conn, string name, int price, string specialist) (додаток Б.3.28)	Виконує запит до бази даних на додання нового запису у таблицю дронів

Продовження таблиці 3.2

1	2	3	4
DataBase	Модуль роботи з базою даних	public string[] NewDroneComposition(NpgsqlConnection conn, string drone, string component, int price) (додаток Б.3.29)	Виконує запит до бази даних на додання нового запису у таблицю складу дронів
		public string[] NewSale(NpgsqlConnection conn, string buyer, string drone, int amount, int price, string date, string manager) (додаток Б.3.30)	Виконує запит до бази даних на додання нового запису у таблицю продаж
		public string GetUser(NpgsqlConnection conn, int id) (додаток Б.3.31)	Виконує запит до бази даних на отримання даних користувача за його айді
		public int Autor(NpgsqlConnection conn, string name, string pass) (додаток Б.3.32)	Виконує запит до бази даних на авторизацію користувача
		public int UsernameChecker(NpgsqlConnection conn, string name) (додаток Б.3.33)	Виконує запит до бази даних з метою перевірки заданого ім'я користувача
		public string NewUser(NpgsqlConnection conn, string name, string pass, string post) (додаток Б.3.34)	Виконує запит до бази даних на реєстрацію користувача
		public static NpgsqlConnection GetDBConnection(string host, int port, string database, string username, string password) (додаток Б.3.35)	Встановлює параметри підключення до бази даних
		public static NpgsqlConnection GetDBConnection() (додаток Б.3.36)	Встановлює параметри підключення до локальної бази даних

3.4 Проктування бази даних

Схема бази даних наведена на рисунку 3.11.

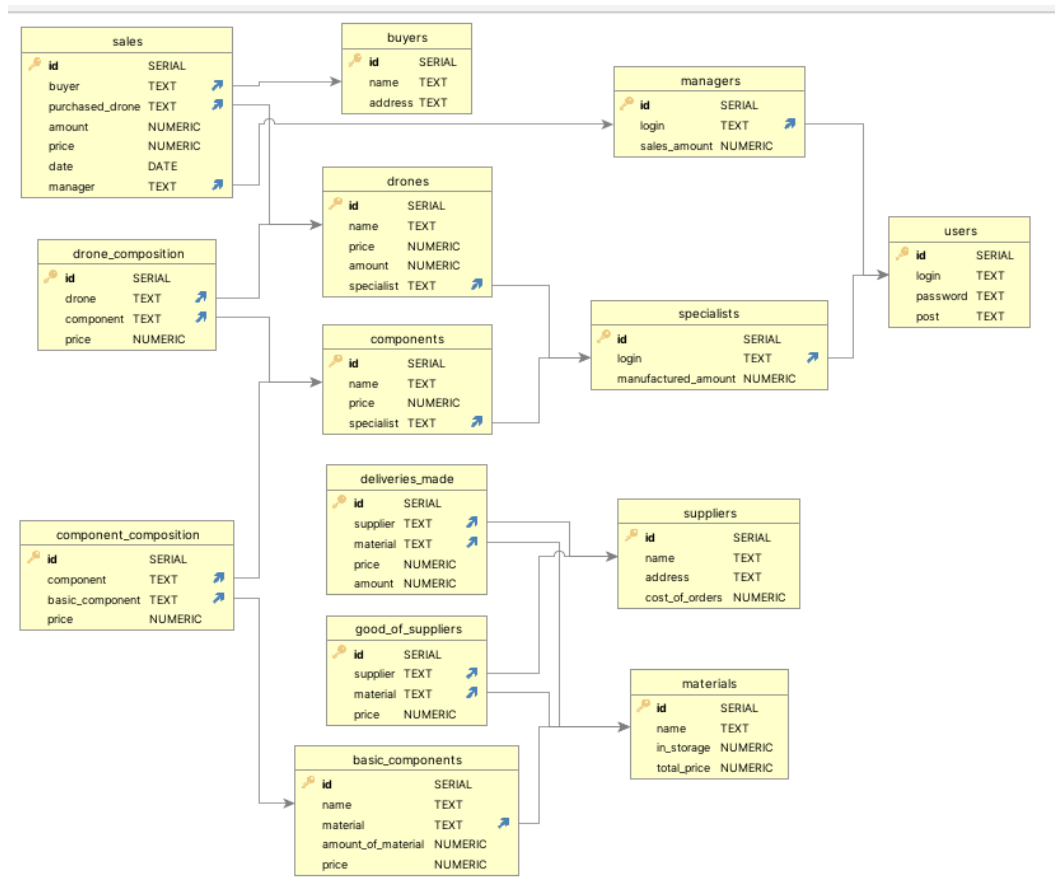


Рисунок 3.11 – Схема бази даних

Сутність materials (id, name, in_storage, total_price) – описує матеріал виготовлення базових компонентів. Опис полів сутності та їх типу наведено в табл. 3.3.

Таблиця 3.3 – Опис полів сутності materials

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	in_storage	NUMERIC
4	total_price	NUMERIC

Сутність suppliers (id, name, address, cost_of_orders) – описує постачальників товарів. Опис полів сутності та їх типу наведено в табл. 3.4.

Таблиця 3.4 – Опис полів сутності suppliers

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	address	TEXT
4	cost_of_orders	NUMERIC

Сутність deliveries_made (id, supplier, material, price, amount) – описує зроблені постачання. Опис полів сутності та їх типу наведено в табл. 3.5.

Таблиця 3.5 – Опис полів сутності deliveries_made

№	Назва поля	Тип поля
1	id	SERIAL
2	supplier	TEXT
3	material	TEXT
4	price	NUMERIC
5	amount	NUMERIC

Сутність good_of_suppliers (id, supplier, material, price) – описує товари постачальників. Опис полів сутності та їх типу наведено в табл. 3.6.

Таблиця 3.6 – Опис полів сутності good_of_suppliers

№	Назва поля	Тип поля
1	id	SERIAL
2	supplier	TEXT
3	material	TEXT
4	price	NUMERIC

Сутність basic_components (id, name, material, amount_of_material, price) – описує базові компоненти. Опис полів сутності та їх типу наведено в табл. 3.7.

Таблиця 3.7 – Опис полів сутності basic_components

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	material	TEXT
4	amount_of_material	NUMERIC
5	price	NUMERIC

Сутність component_composition (id, component, basic_component, price) – описує склад компонентів. Опис полів сутності та їх типу наведено в табл. 3.8.

Таблиця 3.8 – Опис полів сутності component_composition

№	Назва поля	Тип поля
1	id	SERIAL
2	component	TEXT
3	basic_component	TEXT
4	price	NUMERIC

Сутність components (id, name, price, specialist) – описує компоненти, їх ціну та спеціаліста який їх створив. Опис полів сутності та їх типу наведено в табл. 3.9.

Таблиця 3.9 – Опис полів сутності components

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	price	NUMERIC
4	specialist	TEXT

Сутність specialists (id, name, manufactured_amount) – описує спеціалістів з виготовлення дронів. Опис полів сутності та їх типу наведено в табл. 3.10.

Таблиця 3.10 – Опис полів сутності specialists

№	Назва поля	Тип поля
1	id	SERIAL
2	login	TEXT
3	manufactured_amount	NUMERIC

Сутність managers (id, name, sales_amount) – описує менеджерів з продаж. Опис полів сутності та їх типу наведено в табл. 3.11.

Таблиця 3.11 – Опис полів сутності managers

№	Назва поля	Тип поля
1	id	SERIAL
2	login	TEXT
3	sales_amount	NUMERIC

Сутність users (id, login, password, post) – описує користувачів системи. Опис полів сутності та їх типу наведено в табл. 3.12.

Таблиця 3.12 – Опис полів сутності users

№	Назва поля	Тип поля
1	id	SERIAL
2	login	TEXT
3	password	TEXT
4	post	TEXT

Сутність drones (id, name, price, amount, specialist) – описує дронів. Опис полів сутності та їх типу наведено в табл. 3.13.

Таблиця 3.13 – Опис полів сутності drones

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	price	NUMERIC
4	amount	NUMERIC
5	specialist	TEXT

Сутність drone_composition (id, drone, component, price) – описує склад дронів. Опис полів сутності та їх типу наведено в табл. 3.14.

Таблиця 3.14 – Опис полів сутності drone_composition

№	Назва поля	Тип поля
1	id	SERIAL
2	drone	TEXT
3	component	TEXT
4	price	NUMERIC

Сутність buyers (id, name, address) – описує покупців. Опис полів сутності та їх типу наведено в табл. 3.15.

Таблиця 3.15 – Опис полів сутності buyers

№	Назва поля	Тип поля
1	id	SERIAL
2	name	TEXT
3	address	TEXT

Сутність sales (id, buyer, purchased_drone, amount, price, date, manager) – описує продажі. Опис полів сутності та їх типу наведено в табл. 3.16.

Таблиця 3.16 – Опис полів сутності sales

№	Назва поля	Тип поля
1	id	SERIAL
2	buyer	TEXT
3	purchased_drone	TEXT
4	amount	NUMERIC
5	price	NUMERIC
6	date	DATE
7	manager	TEXT

3.5 Висновки за розділом 3

Як результат виконаної роботи, у третьому розділі було описано основні алгоритми розроблюваної програми. За допомогою діаграм було продемонстровано проектування та архітектуру розробленої системи. Було наведено та описано інформацію про модулі системи, а також наведено схему бази даних та описано її сутності.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Програма призначена для застосування під час обліку виробництва підприємства з виготовлення дронів.

4.2 Інструкція по роботі з програмою

4.2.1 Встановлення програми

Для встановлення програми необхідно завантажити на комп'ютер клієнтський застосунок формату .exe та серверний застосунок, що складається з файлу формату .exe та бібліотек. Далі необхідно додати базу даних до СКБД PostgreSQL встановленій на серверній частині системи.

4.2.2 Запуск програми

Запуск для клієнтської частини системи виконується через файл Drone_ClientApp.exe, а серверної частини через файл Drone_ServerApp.exe.

4.2.3 Методика роботи з програмою

Для роботи з програмою спочатку необхідно запустити серверну частину програми на локальному або іншому сервері. Після цього слід запустити клієнтську частину програми та встановити параметри підключення до сервера. Далі необхідно пройти процедуру аторизації або реєстрації, після чого користувача надається можливість працювати з відповідними для його ролі в системі даними.

4.3 Приклад роботи користувача з програмою

Робота програми починається з запуску сервера (рис. 4.1).

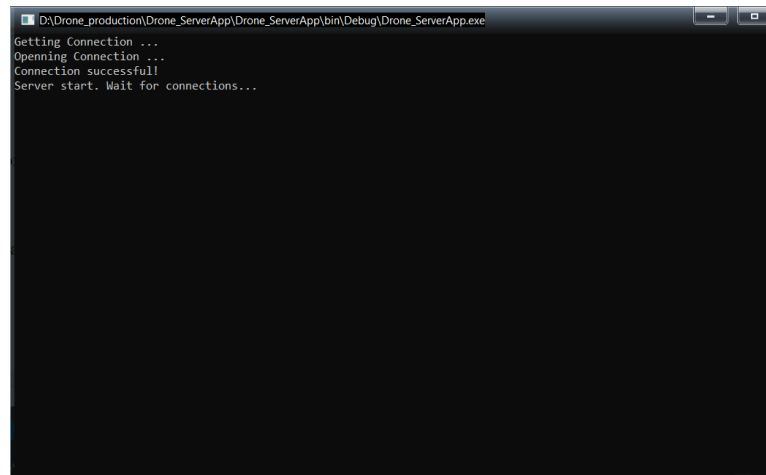


Рисунок 4.1 – Початок роботи сервера

Після цього слід запустити клієнтський застосунок. Форма з якої починається робота програми – це графічна форма встановлення параметрів підключення (рис. 4.2). Якщо натиснути на «Local server», то параметри підключення будуть встановлені за замовчуванням (IP: 127.0.0.1 та порт: 8080). При натисканні на «Public server» форма зміниться для вводу даних (рис. 4.3). Після натисканні на кнопки «Connect» або «Local server» користувача перекине на форму авторизації.

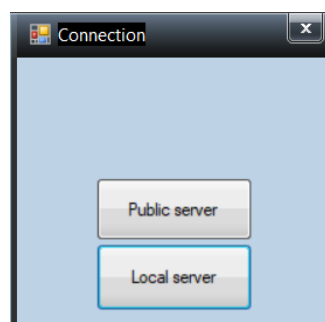


Рисунок 4.2 – Форма встановлення параметрів підключення. Початковий стан

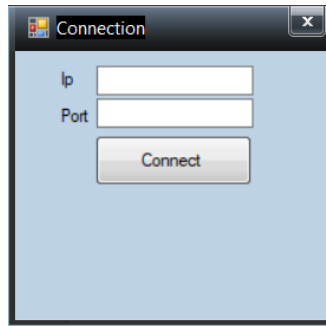


Рисунок 4.3 – Форма встановлення параметрів підключення

На формі авторизації клієнт має можливість ввести у текстові поля «Name» та «Password» свої ім'я та пароль (рис. 4.4). Після натискання на кнопку «Login» програма зробе спробу авторизації, і якщо спроба виявиться вдалою – переведе користувача на графічну форму навігації, у іншому випадку – ні. Якщо натиснути кнопку «Start Registration», форма змине свій вигляд під форму реєстрації, додавши поле вибору ролі. При виборі ролі та натисканні на кнопку «Registration» програма зробе спробу реєстрації користувача, у випадку вдачі, переведе користувача на форму його ролі, у іншому випадку – ні (рис. 4.5).

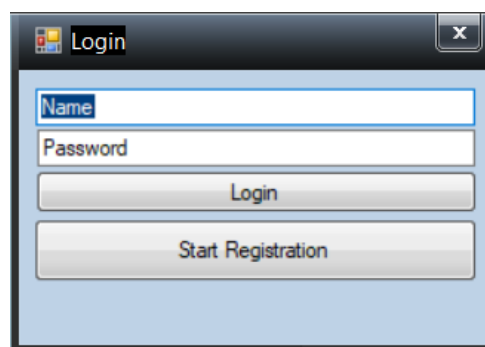


Рисунок 4.4 – Форма системи авторизації

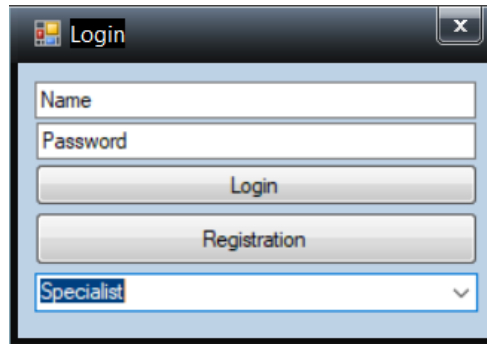
A screenshot of a 'Login' window. It has a title bar with a logo and the text 'Login'. Inside, there are two input fields: 'Name' and 'Password'. Below them are two buttons: 'Login' and 'Registration'. At the bottom is a dropdown menu with 'Specialist' selected.

Рисунок 4.5 – Форма авторизації. Реєстрація

Далі відкривається графічна форми навігації, на якій в залежності від ролі користувача будуть кнопки переходу на графічну форму його ролі та форму формування звітів (рис. 4.6 – 4.8).

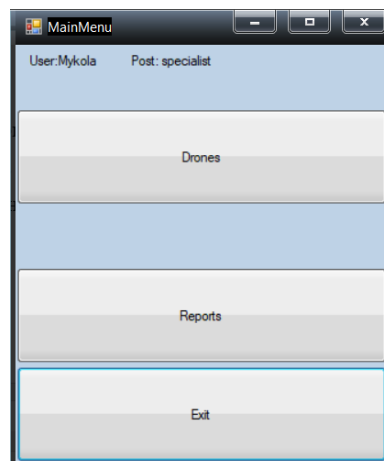
A screenshot of a 'MainMenu' window. The title bar says 'MainMenu'. Below the title bar, it displays 'User: Mykola' and 'Post: specialist'. There are three buttons stacked vertically: 'Drones', 'Reports', and 'Exit'. The 'Exit' button is highlighted with a blue border.

Рисунок 4.6 – Графічна форма навігації. Варіант для спеціаліста

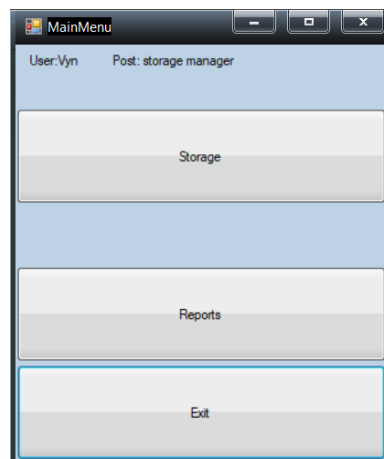
A screenshot of a 'MainMenu' window. The title bar says 'MainMenu'. Below the title bar, it displays 'User: Vyn' and 'Post: storage manager'. There are three buttons stacked vertically: 'Storage', 'Reports', and 'Exit'. The 'Exit' button is highlighted with a blue border.

Рисунок 4.7 – Графічна форма навігації. Варіант для складського менеджера

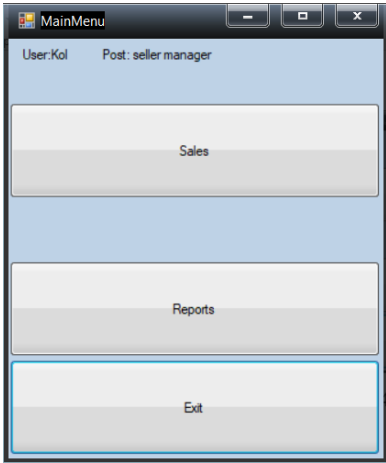


Рисунок 4.8 – Графічна форма навігації. Варіант для менеджера з продаж

Якщо роль користувача в системі – менеджер з продаж, то при натисканні кнопки «Sales» на графічній формі навігації йому відкриється форма роботи з продажами (рис. 4.9). На даній формі є три списки: у лівій частині вікна – покупців, у правій – список дронів, а у нижній частині форми знаходиться список здійснених продаж. Також на формі є дві групи елементів: додання нового покупця та здійснення продажі. При натисканні на кнопку «New buyer» на сервер буде надісланий запит на додання нового покупця із заданими у полях даними. При натисканні на кнопку «New sale» на сервер буде надісланий запит на додання нової продажі із заданими у полях даними.

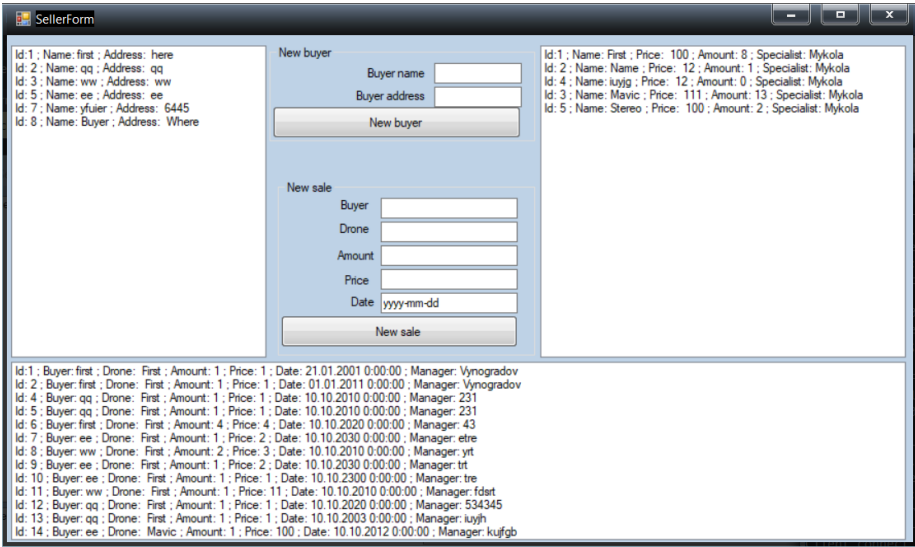


Рисунок 4.9 – Графічна форма менеджера з продаж

Якщо роль користувача в системі – спеціаліст з виготовлення дронів, то при натисканні кнопки «Drones» на графічній формі навігації йому відкриється форма роботи з дронами (рис. 4.10). На даній формі є шість списків даних: у правому верхньому кутку – список матеріалів, посередині зверху – базові компоненти, зліва зверху – список компонентів, зліва знизу – склад компонентів, посередині знизу – список дронів, у правому нижньому кутку – склад дронів.

Також на даній графічній формі є шість груп елементів:

- «New material» з кнопкою та полем для здійснення запиту на сервер з метою додання нового матеріалу в базу даних;
- «New basic component» з полями для введення даних та кнопкою для здійснення запиту до сервера на додання нового базового компоненту;
- «New component» з полями для введення даних та кнопкою для здійснення запиту до сервера на додання нового компонента. У випадку наявності компонента з такою назвою – виведе відповідне повідомлення;
- «New component composition» з полями для введення даних та кнопкою, що здійснює запит до сервера на додання до складу компонента нового базового компонента;
- «New drone» з полями для введення даних та кнопкою «Add new drone» для здійснення запиту до сервера на додання нового дрона та «Add drone» для додання одного заданого дрона;
- «New drone composition» з полями для введення даних та кнопкою, що здійснює запит до сервера на додання до складу дрона нового компонента.

Рисунок 4.10 – Графічна форма спеціаліста з виготовлення дронів

Якщо роль користувача в системі – складського менеджера, то при натисканні кнопки «Storage» на графічній формі навігації йому відкриється форма роботи із закупівлями (рис. 4.11). На даній формі є три списки даних: у правій частині форми – список матеріалів, у верхній лівій – список постачальників, а у нижній лівій – список товарів постачальників. Також на даній графічній формі є три групи елементів:

- «New supplier» з полями для введення даних та кнопкою для здійснення запиту на сервер для додання нового постачальника у базу даних;
- «New supplier goods» з полями для введення даних та кнопкою для здійснення запиту на сервер для додання нового товару постачальника у базу даних;
- «Make Delivery» з полями для введення даних та кнопкою для здійснення запиту на сервер для додання нового постачання у базу даних.

Рисунок 4.11 – Графічна форма складського менеджера

В незалежності від ролі користувач має доступ до графічної форми створення звітів натиснуваши «Reports» на навігаційній формі (рис. 4.12). На даній форми є чотири кнопки:

- «Report deliveries» створює звіт з поставчань;
- «Report drones» створює звіт з дронів на складі;
- «Report sales» створює звіт з продаж дронів;
- «Calculate money from drone selling» розраховує кількість грошей отриманих від продажу дронів.

Рисунок 4.12 – Графічна форма створення звітів

4.4 Тестування роботи програми

Тестування програми буде проводитися на персональному комп'ютері з процесором Intel Pentium CPU N4200, оперативною пам'яттю 4 Гб, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи Microsoft Windows 10.

Тестування буде проведене за таким сценарієм: спочатку проведення тестування форми встановлення параметрів підключення. Після цього – функцій графічної форми авторизації. Далі буде проведене тестування функцій інших графічних форм:

- навігаційної форми. Функцій відкриття графічних форм спеціаліста, складського менеджера та менеджера з продаж. Відкриття графічної форми звітів;
- складського менеджера. Функції додавання нового постачальника. Додавання нового товару постачальника. Здійснення постачання;
- спеціаліста з виготовлення дронів. Функцій додавання нового матеріалу, базового компонента, компонента, дрона. Додавання нового базового компонента до складу компонента. Додавання нового компоненту до складу дрона. Додавання на склад одного дрона;
- графічна форма звітів. Функцій створення звітів з продаж, дронів на складі, здійснених постачань. Розрахунок вартості всіх проданих дронів;
- менеджера з продаж. Функцій додавання нового покупця, здійснення продажі дрона.

Результат тестування розробленого програмного забезпечення наведено в табл. 4.1.

Таблиця 4.1 – Тестування програми

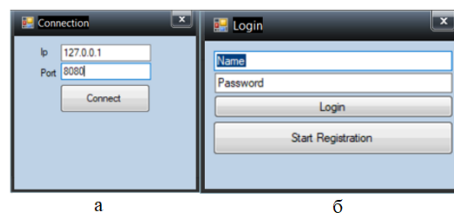
№	Функція	Результат
1	2	3
1	Встановлення параметрів підключення.	Параметри підключення успішно встановлено.
2	Авторизація користувача.	Користувач пройшов авторизацію.
3	Реєстрація користувача. Відкриття навігаційної форми.	Користувач зареєстрований у системі. Навігаційна форма відкрита.
4	Відкриття графічної форми спеціаліста з виготовлення дронів. Завантаження даних.	Графічна форма спеціаліста відкрита. Дані успішно завантажені.
5	Форма спеціаліста. Додання нового матеріалу.	Новий матеріал був успішно доданий.
6	Форма спеціаліста. Додання нового базового компонента.	Новий базовий компонент був успішно доданий.
7	Форма спеціаліста. Додання нового компонента.	Новий компонент був успішно доданий.
8	Форма спеціаліста. Додання нового базового компонента у склад компонента.	Новий базовий компонент був успішно доданий до складу компонента.
9	Форма спеціаліста. Додання нового дрона.	Новий дрон був успішно доданий.
10	Форма спеціаліста. Додання одного обраного дрона.	Новий обраний дрон був успішно доданий.
11	Форма спеціаліста. Додання нового компонента у склад дрона.	Новий компонент у склад дрона був успішно доданий.
12	Відкриття графічної форми менеджера з продаж. Завантаження даних.	Графічна форма менеджера з продаж відкрита. Дані успішно завантажені.
13	Форма менеджера з продаж. Додання нового покупця.	Новий покупець був успішно доданий.
14	Форма менеджера з продаж. Додання нової продажі.	Нова продаж була успішно додана.
15	Відкриття графічної форми складського менеджера. Завантаження даних.	Графічна форма складського менеджера відкрита. Дані успішно завантажені.
16	Форма складського менеджера. Додання нового постачальника.	Новий постачальник був успішно доданий.

Продовження таблиці 4.1

1	2	3
17	Форма складського менеджера. Додання нового товару постачальника.	Новий товар постачальника був успішно доданий.
18	Форма складського менеджера. Виконання нового постачання.	Новий нове постачання було успішно додано до бази даних.
19	Відкриття графічної форми звітів.	Форма звітів відкрита.
20	Формування звіту з дронів на складі.	Звіт з дронів успішно сформований.
21	Формування звіту з постачань.	Звіт з постачань успішно сформований
22	Формування звіту з продаж та розрахунок суми отриманої за продаж всіх дронів	Звіт з продаж успішно сформований. Сума була розрахована.

Тестування функцій програми:

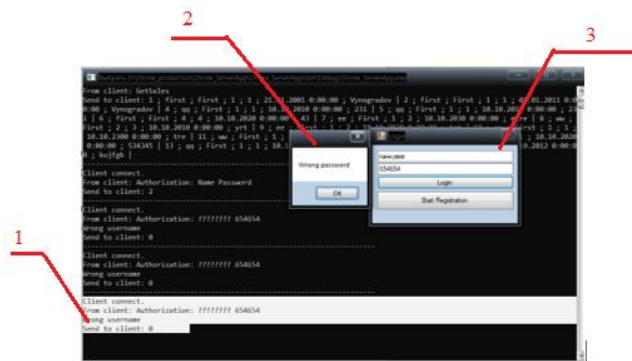
— встановлення параметрів підключення за замовчуванням та підключення до публічного серверу. Оскільки обидва способи працюють за подібними алгоритмами (встановити значення змінних підключення) для отримання результату тестування підійде тестування підключення до публічного серверу. Оскільки форма встановлення параметрів підключення закрилася, а форма авторизації відкрилася – встановлення параметрів було проведене успішно. Результат продемонстровано на рис. 4.13;



а – встановлення параметрів підключення; б – відкриття форми авторизації

Рисунок 4.13 – Встановлення параметрів підключення, після натискання кнопки – відкриття форми авторизації

– авторизація користувача. Для перевірки функцій авторизації, введемо у відповідні поля відкритої форми авторизації дані – ім'я та пароль. Якщо користувач буде знайдений – сервер передасть параметр відповідний за роль користувача у системи, а потім відкриється навігаційна форма. У випадку, якщо користувача не було знайдено, або він ввів некоректний пароль – на стороні серверу видасть повідомлення про відсутність такого користувача у системі. На рис. 4.14 продемонстровано вхідні дані введені користувачем до форми авторизації, та реакцію серверу на різні вхідні дані.



1 – реакція сервера; 2 – реакція на неправильно введений пароль; 3 – Форма авторизації

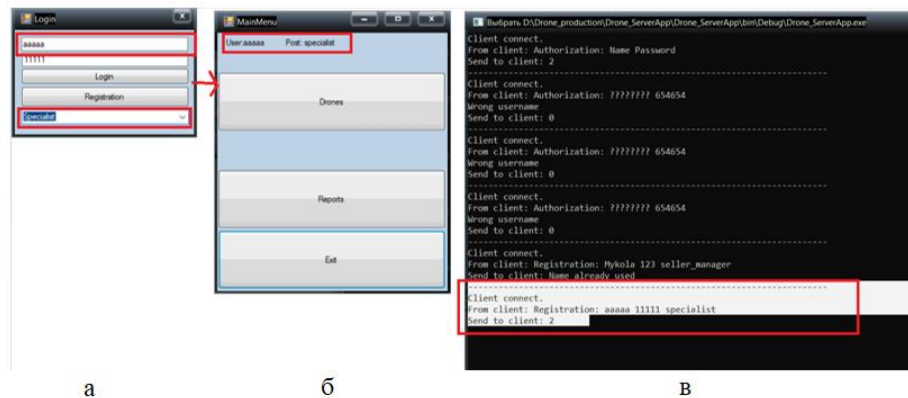
Рисунок 4.14 – Авторизація користувача

– реєстрація користувача. На рис. 4.15 можна побачити реакцію програми на реєстрацію вже існуючого користувача. На рис. 4.16 показано дані введені для тестування реєстрації нового користувача та відкриту форму після реєстрації;



1 – Реакція сервера; 2 – Реакція фомри авторизації на існуючого користувача

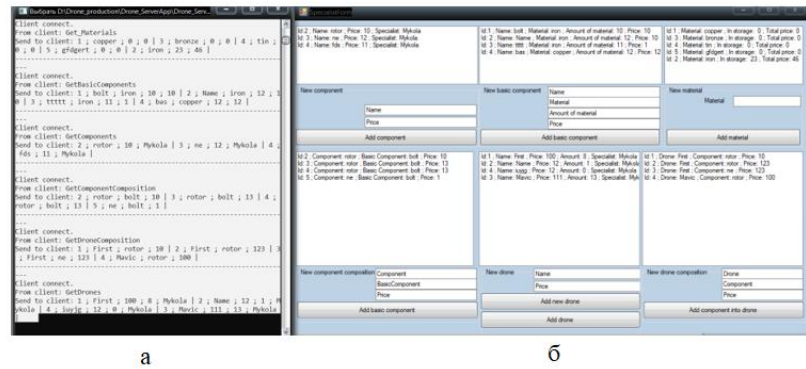
Рисунок 4.15 – Реєстрація існуючого користувача



а – Введення даних у форму атворизації; б – відкриття форми навігації; в – реакція сервера

Рисунок 4.16 – Реєстарція користувача та відкриття форми навігації

– під відкриття графічної форми спеціаліста з виготовлення дронів виконуються запитів на сервер для завантаження даних для списків на формі. Результат продемонстровано на рис. 4.17;



а – реакція сервера на запит завантаження даних; б – форма спеціаліста із завантаженими даними

Рисунок 4.17 – Відкриття форми спеціаліста

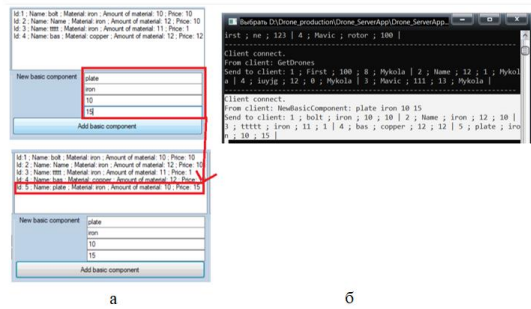
– форма спеціалісту. Додання нового матеріалу. Для тестування даного функціоналу необхідно ввести в поле на групі елементів «New material» назву нового матеріалу після чого натиснути на кнопку. Результат продемонстровано на рис. 4.18;



а – введення даних; б – дані додані; в – реакція сервера

Рисунок 4.18 – Додання нового матеріалу

– форма спеціаліста. Додання нового базового компонента. Для виконання даного тестування необхідно в текстові поля групи «New basic component» ввести дані та натиснути на кнопку. Результат продемонстровано на рис. 4.19;



а – введення і додання даних; б – реакція сервера

Рисунок 4.19 – Додання нового базового компонента

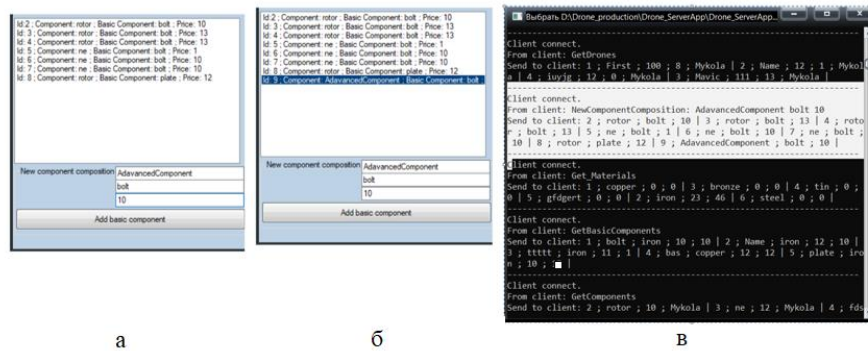
– форма спеціаліста. Додання нового компонента. Для виконання даного тестування необхідно в текстові поля групи «New component» ввести дані та натиснути на кнопку. Результат продемонстровано на рис. 4.20;



а – введення даних; б – дані успішно додані; в – реакція сервера

Рисунок 4.20 – Додання нового компонента

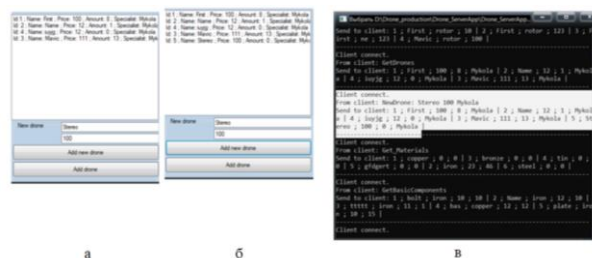
– форма спеціаліста. Додання нового базового компонента у склад дрона. Для виконання тестування необхідно у групі елементів «New component composition» ввести у текстові поля необхідні дані та натиснути на кнопку . Результат продемонстровано на рис. 4.21;



а – введення даних; б – дані успішно додані; в – реакція сервера

Рисунок 4.21 – Додання до складу компонента базового компонента

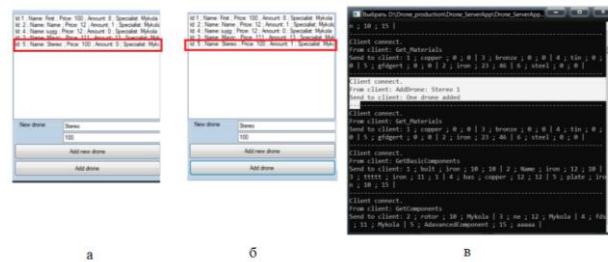
– форма спеціаліста. Додання нового дрона. Для виконання тестування необхідно у групі елементів «New drone» ввести у текстові поля необхідні дані та натиснути на кнопку. Результат продемонстровано на рис. 4.22;



а – введення даних; б – дані успішно додані; в – реакція сервера

Рисунок 4.22 – Додання нового дрона

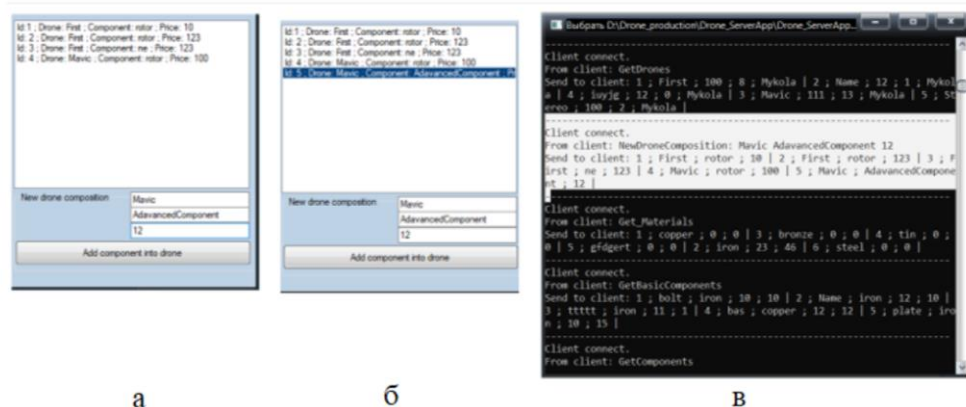
– форма спеціаліста. Додання одного обраного дрона. Для виконання тестування необхідно у групі елементів «New drone» ввести у текстові поля необхідні дані та натиснути на кнопку «Add drone». Результат продемонстровано на рис. 4.23;



а – введення даних; б – дані успішно додані; в – реакція сервера

Рисунок 4.23 – Додання одного дрона

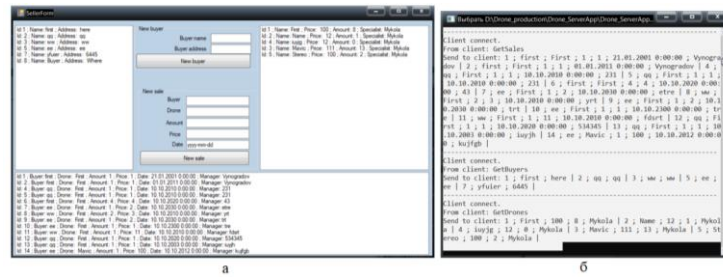
– форма спеціаліста. Додання нового компонента у склад дрона. Для виконання тестування необхідно у групі елементів «New drone composition» ввести у текстові поля необхідні дані та натиснути на кнопку. Результат продемонстровано на рис. 4.24;



а – введення даних; б – дані успішно додані; в – реакція сервера

Рисунок 4.24 – Додання нового компонента у склад дрона

– відкриття графічної форми менеджера з продаж та завантаження даних. Під час завантаження форми програма надсилає на сервер запити для отримання даних для списків. Результат продемонстровано на рис. 4.25;



а – Відкриття форми; б – реакція сервера

Рисунок 4.25 – Відкриття форми менеджера з продаж

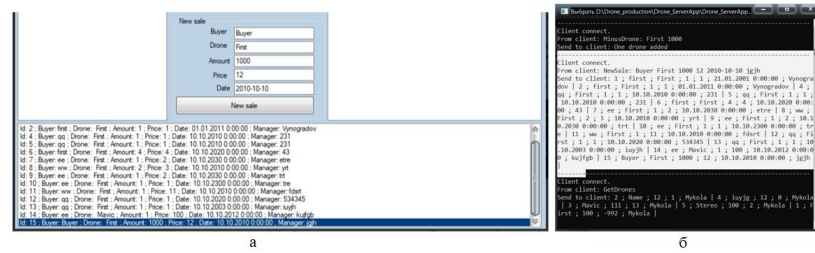
– форма менеджера з продаж. Додання нового покупця. Для виконання тестування необхідно у групі елементів «New buyer» ввести у текстові поля необхідні дані та натиснути на кнопку. Результат продемонстровано на рис. 4.26;



а – введення даних; б – дані успішно додані; в– реакція сервера

Рисунок 4.26 – Додання нового покупця

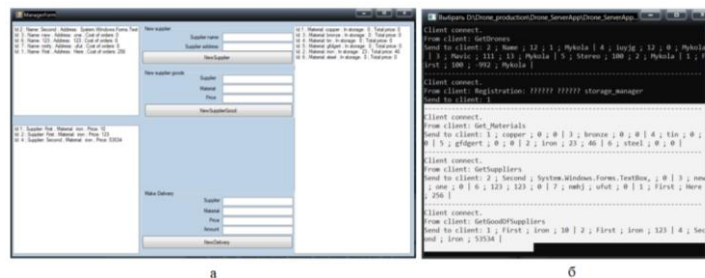
– форма менеджера з продаж. Додання нової продажі. Для виконання тестування необхідно у групі елементів «New sale» ввести у текстові поля необхідні дані та натиснути на кнопку. Результат продемонстровано на рис. 4.27;



а – введення даних та їх додання; б – реакція сервера

Рисунок 4.27 – Додання нової продажі

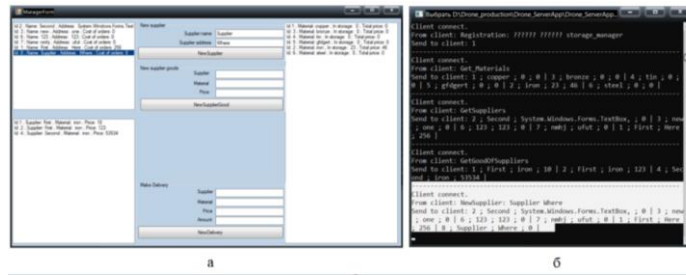
– відкриття графічної форми складського менеджера та завантаження даних. Під час завантаження форми програма надсилає на сервер запити для отримання даних для списків. Результат продемонстровано на рис. 4.28;



а – відкриття форми та завантаження даних; б – реакція сервера

Рисунок 4.28 – Відкриття форми складського менеджера та завантаження даних

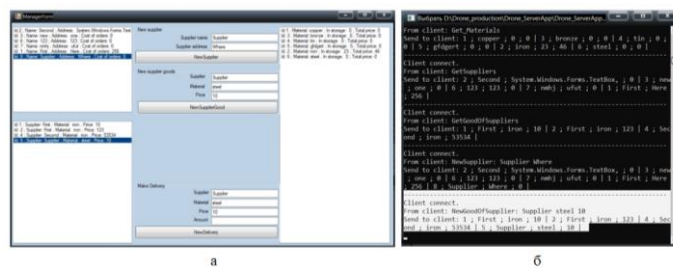
– форма Складського менеджера. Додання нового постачальника. Для виконання тестування даного функціоналу необхідно у текстові поля групи «New supplier» дані та натиснути на кнопку. Результат продемонстровано на рис. 4.29;



а – введення та додання даних; б – реакція сервера

Рисунок 4.29 – Додання нового постачальника

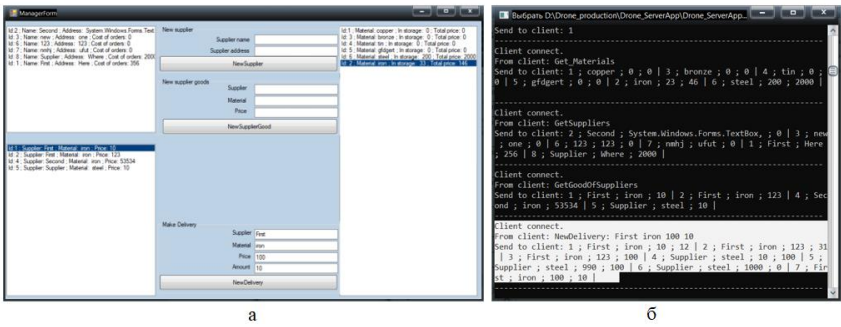
– форма Складського менеджера. Додання нового товару постачальника. Для виконання тестування даного функціоналу необхідно у текстові поля групи «New supplier goods» дані та натиснути на кнопку. Результат продемонстровано на рис. 4.30;



а – введення та додання даних; б – реакція сервера

Рисунок 4.30 – Додання нового товару постачальника

– форма складського менеджера. Виконання нового постачання. Для виконання тестування необхідно у текстові поля групи елементів «Make delivery» ввести необхідні дані, або обрати їх зі списків та натиснути на кнопку. Результат продемонстровано на рис. 4.31;



а – введення та додання даних; б – реакція сервера

Рисунок 4.31 – Виконання нового постачання

– відкриття графічної форми звітів. Для виконання даного тестування необхідно з форми навігації натиснувши кнопку «Reports» відкрити графічну форму звітів. Результат продемонстровано на рис. 4.32;

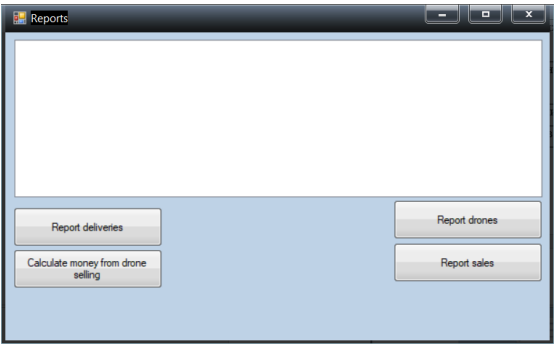


Рисунок 4.32 – Відкриття форми звітів

– формування звіту з дронів на складі. Для цього необхідно натиснути конпку «Report drones» Виконання даного функціоналу продемонстровано на рис. 4.33;

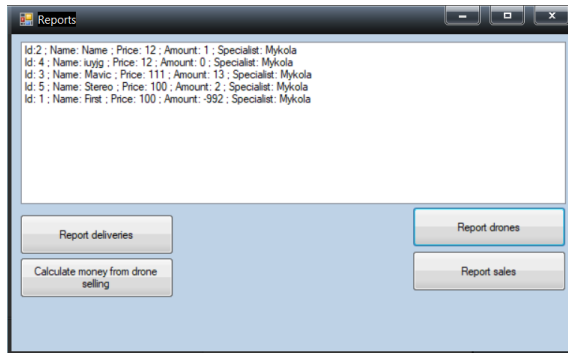


Рисунок 4.34 – Звіт з дронів на складі

– формування звіту з поставач. Для цього необхідно натиснути кнопку «Report deliveries» Виконання даного функціоналу продемонстровано на рис. 4.35;

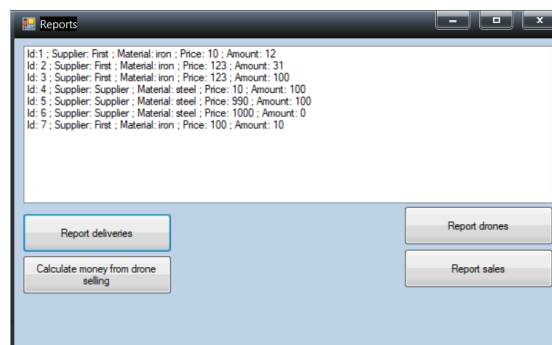
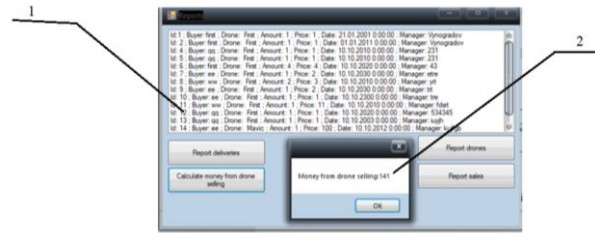


Рисунок 4.35 – Звіт з поставач

– формування звіту з продаж та розрахунок суми отриманої за продаж всіх дронів. Для виконання даного тестування необхідно спочатку натиснути кнопку «Report Sales» а потім «Calculate money from drone selling». Результат продемонстровано на рис. 4.36;



1 – сформований звіт; 2 – розрахунок суми

Рисунок 4.36 – Звіт з продаж та розрахунок суми

ВИСНОВКИ

У роботі вирішено актуальне завдання розроблення програмної реалізації системи обліку виробництва дронів. Було виконане ознайомлення із найбільш важлими питаннями даної предметної області та найвагомішими рішеннями: розподілення ролей користувачів, оптимізація процесів обліку, зберігання інформації у базі даних.

- під час аналізу предметної області було розглянуто існуючі аналоги та виявлено їх недоліки та переваги й сформовано завдання до подальшої роботи;

- була розроблена архітектура реалізованої системи, описано її структуру та функціонування. Після цього було обрано та обґрунтовано мову програмування, середовище розробки та систему керування базами даних. Поставлено вимоги до технічних і програмних засобів. Було наведено концептуальну модель бази даних у вигляді схеми та описано її сутності;

- було розроблено програмний продукт як реалізував всі поставлені задачі. Було описано алгоритм його функціонування, наведено діаграми класів та описано модулі розробленої системи. Зокрема, було наведено розроблену схему бази даних та описано її сутності;

- тестування програми довело, що розроблена система працює правильно, відповідає всім поставленим вимогам та виконує весь необхідний функціонал.

Реалізоване програмне забезпечення доцільно використовувати на підприємствах з виробництва дронів для обліку виробництва з метою підвищення ефективності та оптимізації процесів роботи з даними.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. HugeProfit – Асоціація стартапів України [Електрон. ресурс] – Режим доступу: <https://startups.org.ua/portfolio/hugeprofit/>.
2. Довідка програми УкрСклад [Електрон. ресурс] – Режим доступу: https://www.softbalance.com.ua/help/ukrsklad_ua.html.
3. Ураїнська програма обліку | Торгсофт [Електрон. ресурс] – Режим доступу: https://www.softbalance.com.ua/help/ukrsklad_ua.html.
4. What is Client-Server Architecture? Definition, Types, & Example [Electronic resource]. – Acces mode: <https://www.theknowledgeacademy.com/blog/client-server-architecture/>.
5. Harry, H. C. C# programming : the ultimate way to learn the fundamentals of the C# language [Text] / H. C. Harry. – М. : Createspace Inc, 2014. – 7 p.
6. C# - Basic syntax [Electronic resource]. – Access mode: https://www.tutorialspoint.com/csharp/csharp_basic_syntax.htm.
7. Динамічна типізація в C# [Електрон. ресурс]. – Режим доступу: <https://medium.com/@fightermanlviv/%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D1%96%D1%87%D0%BD%D0%B0-%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%B2-c-e94367da8d91>.
8. Using threads and threading - .NET | Microsoft Learn [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/standard/threading/using-threads-and-threading>.
9. In-Depth – Memory Management in C# | Prographers [Electronic resource]. – Access mode: <https://prographers.com/blog/in-depth-memory-management-in-c>.
10. Standard library – C# language specification | Microsoft Lean [Electronic resource]: Acces mode – <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/standard-library>.

11. Introduction to Visual Studio [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-visual-studio/>.

12. Visual Studio IDE with .Net – Develop Any App Using C#, F#, VB [Electronic resource]: Access mode – <https://visualstudio.microsoft.com/vs/features/net-development/>.

13. Принципи роботи у середовищі візуальної розробки програм [Електрон. ресурс]. – Режим доступу: <https://stboinf.wordpress.com/2013/03/13/%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D0%B8-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B8-%D1%83-%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D1%96-%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%8C/>.

14. Visual Studio – Debugging in Visual Studio | visual-studio Tutorial [Electronic resource]. – Access mode: <https://riptutorial.com/visual-studio/learn/100007/debugging-in-visual-studio>.

15. Find, install, & manage extension packages - Visual Studio (Windows) | Microsoft Learn [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/ide/finding-and-using-visual-studio-extensions?view=vs-2022>.

16. Ocket, J. du P. Visual studio 2022 in-depth : explore the fantastic features of visual studio 2022! [Text] / J. du P. Ocket. – London : BPB Online, 2023. – 21 p.

17. Git – Git y Visual Studio [Електрон. ресурс]. – Режим доступу: <https://git-scm.com/book/uk/v2/%D0%94%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA-A%3A-Git-%D0%B2-%D1%96%D0%BD%D1%88%D0%B8%D1%85-%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B0%D1%85-Git-%D1%83-Visual-Studio>.

18. Visual Studio 2015 Preview, Visual Studio Community 2013, Visual Studio 2013 Update 4, and More - Visual Studio Blog [Electronic resource]. – Access mode: <https://devblogs.microsoft.com/visualstudio/visual-studio-2015-preview-visual-studio-community-2013-visual-studio-2013-update-4-and-more/>.

19. PostgreSQL: About [Electronic resource]. – Access mode: <https://www.postgresql.org/about/>.

20. PostgreSQL [Электрон. ресурс]. – Режим доступа: <http://www.postgresqltutorial.com/what-is-postgresql/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмна реалізація системи обліку виробництва дронів призначене для автоматизації облікової діяльності підприємства, що займається виготовленням дронів.

A.1 Підстави до розробки

Підставою до розробки є завдання на дипломну роботу видане 15 квітня 2024 року.

A.2 Призначення розробки

Програмна реалізація системи обліку виробництва дронів призначене для автоматизації процесів роботи з даними, обліку складської діяльності, відображення поточного становища даних відповідно до поставлених завдань.

A.3 Вимоги до програми чи програмного виробу

A.3.1 Вимоги до функціональних характеристик

Початковими даними програми є наступні таблиці бази даних: «Користувачі», «Постачальники», «Товари постачальників», «Здійснені постачання», «Матеріали», «Склад компонента», «Компоненти», «Склад дрона», «Дрони», «Покупці», «Продажі», «Менеджери з продаж», «Спеціалісти».

Вихідними даними є інформація, отримана клієнтською частиною системи як результат виконання його запитів до сервера.

До складу виконуваних функцій належать наступні:

- відправка запиту на сервер та отримання відповіді;
- отримання запиту на сервері;

- взаємодія із базою даних для клієнта та сервера – клієнт формує запит та надсилає його на сервер, сервер отримує запит, виконує його обробку та надсилає до бази даних, а отриманий результат надсилає клієнту;
- функції формування звітів на даних заснованих на отриманих з бази даних;
- вивидення даних, отриманих від сервера на графічні форми користувача;
- авторизація та реєстрація користувачів;
- надання користувачу функціоналу в залежності від його ролі в системі.

Програма повинна підтримувати наступні типи користувачів:

- складський менеджер. Має доступ до даних про матеріали, постачальники та їх товари. Має можливість додати нового постачальника, його товар та постачання;
- менеджер з продаж. Має доступ до даних про дронів на складі, покупців та здійснених продаж. Має можливість додати нового покупця та продажу;
- спеціаліст з виготовлення дронів. Має доступ до даних про матеріали, базові компоненти, компоненти, дронів та їх склад. Може додати нові дані до відповідних таблиць.

A.3.2 Вимоги до надійності

Надійне функціонування програми повинно бути забезпечено замовником у наступній сукупності організаційно-технічних пунктів:

- безперервне постачання живлення технічних пристроїв на яких буде встановлена програма;
- використання ліцензованого програмного забезпечення;
- регулярне виконання вимог захисту інформації.

Наступні збійні ситуації мають оброблятися програмою:

- відсутній зв'язок з базою даних;
- занесення до бази даних некоректних даних;
- неможливість коректного зчитування даних з бази даних.

A.3.3 Умови експлуатації

Кліматичні умови експлуатації, за яких повинні забезпечуватися задані характеристики, повинні задовольняти умовам, які пред'являються до технічних засобів у частині умов їх експлуатації. Кінцевий користувач програми повинен мати навички у роботі з графічним інтерфейсом.

A.3.4 Вимоги до складу параметрів технічних засобів

Як апаратно-технічні засоби для експлуатації даної системи повинен використовуватися комп'ютер з характеристиками не нижче: процесор 1.1 МГц, оперативна пам'ять 256 Мб, відеопам'ять 64 Мб, HDD 128 Гб під управлінням операційної системи Microsoft Windows 10, підключення до інтернету.

A.3.5 Вимоги до інформаційної й програмної сумісності

Інформаційні структури мають бути розроблені згідно до структури вхідних та вихідних даних.

Вихідні коди програми повинні бути реалізовані на мові C#. Для створення програми буде використане інтегроване середовище розробки Microsoft Visual Studio. Для програмного застосунку буде використаний .NET Framework. Для зберігання даних повинна використовуватися СКБД PostgreSQL.

A.3.6 Вимоги до маркування й упакування

Програма поставляється архіві на пошту замовнику у вигляді двох застосунків: клієнтського та серверного.

Вимоги до упакування відсутні, оскільки програмний продукт доставляється за допомогою інтернетз засобів.

A.3.7 Вимоги до транспортування й збереження

Вимоги до транспортування та збереження відсутні.

A.4 Вимоги до програмної документації

Система поставляється разом з пояснювальною запискою та технічним завданням.

A.5 Техніко-економічні показники

Ефективність програмного забезпечення має оцінюватись в часі, який було зекономлено при обліку виробництва підприємства. Програма не потребує жодних економічних впливань та є цілком безкоштовною. Орієнтована економічна ефективність не розраховується.

A.6 Стадії й етапи розробки

Розробка складається з наступних етапів:

- постановка завдання роботи (1 тиждень);
- аналіз предметної області (1-2 тижні);
- розробка архітектури програми (1 тиждень);
- розробка програми (1-2 тижні);

- тестування та експериментальне дослідження програми (1 тиждень);
- оформлення пояснювальної записки та документів до неї (1 тиждень).

A.7 Порядок контролю та приймання

Випробування повинен проводити розробник, у якого має бути повністю перевірений зазначений функціонал програми. Іспити мають бути проведені для 3 типів користувачів: складський менеджер, менеджер з продаж та спеціаліст з виготовлення дронів. Також мають бути перевірені усі можливі збійні ситуації, та їх обробка. Упевнившись у повній працездатності програми, вона передається замовнику.

ДОДАТОК Б
Текст програми

Б.1 RequestHandler.cs

Текст модулю програми «RequestHandler.cs» що відповідає за попередню обробку запиту від клієнта:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;
namespace Drone_ServerApp
{
    class RequestHandler
    {
        public string Request(string dataReceived)
        {
            try
            {
                NpgsqlConnection conn = DataBase.GetDBConnection();
                //redirect request from client to database

                if (dataReceived == "Get_Materials")
                {
                    DataBase DataBase = new DataBase();
                    string[] materials = DataBase.GetMaterials(conn);
                    string mat_for_ret = "";
                    for (int i = 0; i < materials.Length; i++) mat_for_ret += materials[i] + "
| ";
                    return (mat_for_ret);
                }
                if (dataReceived == "GetComponents")
                {
                    DataBase DataBase = new DataBase();
                    string[] data_array = DataBase.GetComponents(conn);
                    string return_array = "";
                    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
                    return (return_array);
                }
            }
        }
    }
}
```

```

if (dataReceived == "GetBasicComponents")
{

    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.GetBasicComponents(conn);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived == "GetComponentComposition")
{

    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.GetComponentComposition(conn);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived == "GetDroneComposition")
{

    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.GetDroneComposition(conn);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived == "GetDrones")
{

    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.GetDrones(conn);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived == "GetBuyers")
{

    DataBase DataBase = new DataBase();

```

```

        string[] data_array = DataBase.GetBuyers(conn);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived == "GetSales")
    {

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.GetSales(conn);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived == "GetGoodOfSuppliers")
    {

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.GetGoodOfSuppliers(conn);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived == "GetDeliveriesMade")
    {

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.GetDeliveriesMade(conn);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived == "GetSuppliers")
    {

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.GetSuppliers(conn);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

```

```

        return (return_array);
    }
    if (dataReceived.Contains("UpdateMaterial: "))
    {
        // a = "UpdateMaterial: " + am + " " + (am * price);
        string[] words = dataReceived.Split(' ');
        string name = words[1];
        string amount = words[2];
        string price = words[3];
        DataBase DataBase = new DataBase();
        DataBase.UpdateMaterials(conn,
name,Int32.Parse(amount),Int32.Parse(price));

        return "Material update";
    }
    if (dataReceived.Contains("NewMaterial:"))
    {

        string[] words = dataReceived.Split(' ');
        string name = words[1];

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.NewMaterial(conn, name);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

        return (return_array);
    }
    if (dataReceived.Contains("NewBasicComponent:"))
    {

        string[] words = dataReceived.Split(' ');
        string name = words[1];
        string material = words[2];
        int amount = Convert.ToInt32(words[3]);
        int price = Convert.ToInt32(words[4]);
        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.NewBasicComponent(conn, name,
material, amount, price);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

        return (return_array);
    }
}

```

```

if (dataReceived.Contains("NewComponent:"))
{
    string[] words = dataReceived.Split(' ');
    string name = words[1];
    int price = Convert.ToInt32(words[2]);
    string specialist = words[3];
    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewComponent(conn, name, price,
specialist);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
    return (return_array);
}
if (dataReceived.Contains("NewComponentComposition:"))
{
    string[] words = dataReceived.Split(' ');
    string component = words[1];
    string basic_component = words[2];
    int price = Convert.ToInt32(words[3]);
    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewComponentComposition(conn,
component, basic_component, price);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
    return (return_array);
}
if (dataReceived.Contains("NewSupplier:"))
{
    string[] words = dataReceived.Split(' ');
    string name = words[1];
    string address = words[2];
    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewSupplier(conn, name, address);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
    return (return_array);
}
if (dataReceived.Contains("NewSale:"))

```

```

{

    string[] words = dataReceived.Split(' ');
    string buyer = words[1];
    string drone = words[2];
    int amount = Convert.ToInt32(words[3]);
    int price = Convert.ToInt32(words[4]);
    string date = words[5];
    string manager= words[6];
    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewSale(conn, buyer,
drone,amount,price,date,manager);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived.Contains("NewBuyer:"))
{

    string[] words = dataReceived.Split(' ');
    string name = words[1];
    string address = words[2];

    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewBuyer(conn, name, address);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

    return (return_array);
}
if (dataReceived.Contains("NewGoodOfSupplier:"))
{

    string[] words = dataReceived.Split(' ');
    string supplier = words[1];
    string material = words[2];
    int price = Convert.ToInt32(words[3]);
    DataBase DataBase = new DataBase();
    string[] data_array = DataBase.NewGoodOfSupplier(conn, supplier,
material, price);
    string return_array = "";
    for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";

```

```

        return (return_array);
    }
    if (dataReceived.Contains("NewDelivery:"))
    {

        string[] words = dataReceived.Split(' ');
        string supplier = words[1];
        string material = words[2];
        int price = Convert.ToInt32(words[3]);
        int amount = Convert.ToInt32(words[4]);
        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.NewDelivery(conn, supplier, material,
price, amount);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived.Contains("NewDrone:"))
    {

        string[] words = dataReceived.Split(' ');
        string name = words[1];
        int price = Convert.ToInt32(words[2]);
        string specialist = words[3];

        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.NewDrone(conn, name, price,
specialist);
        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
        return (return_array);
    }
    if (dataReceived.Contains("NewDroneComposition:"))
    {

        string[] words = dataReceived.Split(' ');
        string drone = words[1];
        string component = words[2];
        int price = Convert.ToInt32(words[3]);
        DataBase DataBase = new DataBase();
        string[] data_array = DataBase.NewDroneComposition(conn, drone,
component, price);

```

```

        string return_array = "";
        for (int i = 0; i < data_array.Length; i++) return_array += data_array[i]
+ " | ";
    return (return_array);
}
if (dataReceived.Contains("AddDrone:"))
{

    string[] words = dataReceived.Split(' ');
    string drone = words[1];
    int amount = Convert.ToInt32(words[2]);
    DataBase DataBase = new DataBase();
    DataBase.UpdateDrones(conn, drone, amount);
    return "One drone added";
}
if (dataReceived.Contains("MinusDrone:"))
{

    string[] words = dataReceived.Split(' ');
    string drone = words[1];
    int amount = Convert.ToInt32(words[2]);
    DataBase DataBase = new DataBase();
    DataBase.UpdateMinusDrones(conn, drone, amount);
    return "One drone added";
}

if (dataReceived.Contains("Authorization:"))
{

    string[] words = dataReceived.Split(' ');
    string name = words[1];
    string password = words[2];

    DataBase DataBase = new DataBase();
    return (DataBase.Autor(conn, name, password).ToString());
}
if (dataReceived.Contains("Registration:"))
{

    string[] words = dataReceived.Split(' ');
    string name = words[1];
    string password = words[2];
    string post = words[3];

```

```

        DataBase DataBase = new DataBase();
        return(DataBase.NewUser(conn, name, password, post));
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
return "Error!";
}
}
}

```

Б.2 Connector.cs

Текст модулю програми «Connector.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;
using System.Net;
using System.Net.Sockets;
namespace Drone_ServerApp
{
    class Connector
    {
        NpgsqlConnection conn = DataBase.GetDBConnection();
        public TcpListener listener;
        public TcpClient client;
        public string ipAddress = "127.0.0.1"; //Server IP
        public int port = 8080;
        public void connect()
        {
            listener = new TcpListener(IPAddress.Parse(ipAddress), port);
            // TcpListener for input handler
            try
            {

```

```

        // Start handler
        listener.Start();
        Console.WriteLine("Server start. Wait for connections...");
        accept();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: " + ex.Message);
    }
    Console.ReadLine();
}

public void accept()
{
    // Client object connect
    client = listener.AcceptTcpClient();
    // Stream from client
    NetworkStream stream = client.GetStream();
    // Buffer for input stream
    byte[] buffer = new byte[1024];
    int bytesRead = stream.Read(buffer, 0, buffer.Length);
    string dataReceived = Encoding.ASCII.GetString(buffer, 0,
bytesRead);
    if (dataReceived != "")
    {
        Console.WriteLine("-----
-----");
        Console.WriteLine("Client connect.");
        Console.WriteLine("From client: " + dataReceived);
        DataSafe sender = new DataSafe();
        // RequestHandler sender = new RequestHandler();
        Send(sender.Safer(dataReceived));
    }
    else accept();
}

public void Send(string sender)
{
    //function for send stream to client
    NetworkStream stream = client.GetStream();
    byte[] responseBuffer = Encoding.ASCII.GetBytes(sender);
    stream.Write(responseBuffer, 0, responseBuffer.Length);
    Console.WriteLine("Send to client: " + sender);
    accept();
}

```

```
    }
}
```

Б.3 DataBase.cs

Б.3.1 Функція GetMaterials

```
public string[] GetMaterials(NpgsqlConnection conn)
{
    List<string> Materials = new List<string>();

    string sql = "SELECT id, name, in_storage, total_price FROM
materials";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();

    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string Name = reader.GetString(1);
                int In_stock = Convert.ToInt32(reader.GetValue(2));
                int Cost = Convert.ToInt32(reader.GetValue(3));

                Materials.Add(ID.ToString() + " ; " + Name + " ; "
+In_stock.ToString()+" ; "+ Cost.ToString());
            }
        }
        else Materials.Add("No items");
    }
    conn.Close();

    return Materials.ToArray();
}
```

```
}
```

Б.3.2 Функція GetComponents

```
public string[] GetComponents(NpgsqlConnection conn)
{
    List<string> Components = new List<string>();
    string sql = "SELECT ID, name, price, specialist FROM
components";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string name = reader.GetString(1);
                int price = Convert.ToInt32(reader.GetValue(2));
                string specialist = reader.GetString(3);

                Components.Add(ID.ToString() + " ; " + name + " ; " +
price.ToString() + " ; " + specialist);
            }
        }
        else Components.Add("No items");
    }
    conn.Close();
    return Components.ToArray();
}
```

Б.3.3 Функція GetBasicComponents

```
public string[] GetBasicComponents(NpgsqlConnection conn)
{
    List<string> BasicComponents = new List<string>();
```

```

        string sql = "SELECT ID, name, material, amount_of_material, price
FROM basic_components";
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader())
        {
            if (reader.HasRows)    //Якщо є рядки
            {
                while (reader.Read())
                {
                    int ID = Convert.ToInt32(reader.GetValue(0));
                    string name = reader.GetString(1);
                    string material = reader.GetString(2);
                    int amount = Convert.ToInt32(reader.GetValue(3));
                    int price = Convert.ToInt32(reader.GetValue(4));

                    BasicComponents.Add(ID.ToString() + " ; " + name + " ; " +
material + " ; " + amount.ToString() + " ; " + price.ToString());
                }
            }
            else BasicComponents.Add("No items");
        }
        conn.Close();
        return BasicComponents.ToArray();
    }

```

Б.3.4 Функція GetBuyers

```

public string[] GetBuyers(NpgsqlConnection conn)
{
    List<string> Buyers = new List<string>();
    string sql = "SELECT ID, name, address FROM buyers";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки

```

```

{
    while (reader.Read())
    {
        int ID = Convert.ToInt32(reader.GetValue(0));
        string name = reader.GetString(1);
        string address = reader.GetString(2);
        Buyers.Add(ID.ToString() + " ; " + name + " ; " + address);
    }
}
else Buyers.Add("No items");
}
conn.Close();
return Buyers.ToArray();
}

```

Б.3.5 Функція GetComponentComposition

```

public string[] GetComponentComposition(NpgsqlConnection conn)
{
    List<string> Component_composition = new List<string>();
    string sql = "SELECT ID, component, basic_component, price
FROM component_composition";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string component = reader.GetString(1);
                string basic_component = reader.GetString(2);
                int price = Convert.ToInt32(reader.GetValue(3));
                Component_composition.Add(ID.ToString() + " ; " +
component + " ; " + basic_component + " ; " + price.ToString());
            }
        }
        else Component_composition.Add("No items");
    }
}

```

```

    conn.Close();
    return Component_composition.ToArray();
}

```

Б.3.6 Функція GetGoodOfSuppliers

```

public string[] GetGoodOfSuppliers(NpgsqlConnection conn)
{
    List<string> Goods = new List<string>();
    string sql = "SELECT ID, supplier, material, price FROM
good_of_suppliers";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string supplier = reader.GetString(1);
                string material = reader.GetString(2);
                int price = Convert.ToInt32(reader.GetValue(3));
                Goods.Add(ID.ToString() + " ; " + supplier + " ; " + material
+ " ; " + price.ToString());
            }
        }
        else Goods.Add("No items");
    }
    conn.Close();
    return Goods.ToArray();
}

```

Б.3.7 Функція GetDeliveriesMade

```

public string[] GetDeliveriesMade(NpgsqlConnection conn)
{
    List<string> Delivieries = new List<string>();

```

```

        string sql = "SELECT ID, supplier, material, price, amount FROM
deliveries_made";
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader())
        {
            if (reader.HasRows)    //Якщо є рядки
            {
                while (reader.Read())
                {
                    int ID = Convert.ToInt32(reader.GetValue(0));
                    string supplier = reader.GetString(1);
                    string material = reader.GetString(2);
                    int price = Convert.ToInt32(reader.GetValue(3));
                    int amount = Convert.ToInt32(reader.GetValue(4));
                    Delivieries.Add(ID.ToString() + " ; " + supplier + " ; " +
material + " ; " + price.ToString() + " ; " + amount.ToString());
                }
            }
            else Delivieries.Add("No items");
        }
        conn.Close();
        return Delivieries.ToArray();
    }

```

Б.3.8 Функція GetDroneComposition

```

public string[] GetDroneComposition(NpgsqlConnection conn)
{
    List<string> DroneComposition = new List<string>();
    string sql = "SELECT ID, drone, component, price FROM
drone_composition";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки

```

```

    {
        while (reader.Read())
        {
            int ID = Convert.ToInt32(reader.GetValue(0));
            string drone = reader.GetString(1);
            string component = reader.GetString(2);
            int price = Convert.ToInt32(reader.GetValue(3));
            DroneComposition.Add(ID.ToString() + " ; " + drone + " ; "
+ component + " ; " + price.ToString());
        }
    }
    else DroneComposition.Add("No items");
}
conn.Close();
return DroneComposition.ToArray();
}

```

Б.3.9 Функція GetSales

```

public string[] GetSales(NpgsqlConnection conn)
{
    List<string> Sales = new List<string>();
    string sql = "SELECT * FROM sales";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string buyer = reader.GetString(1);
                string drone = reader.GetString(2);
                int amount = Convert.ToInt32(reader.GetValue(3));
                int price = Convert.ToInt32(reader.GetValue(4));
                string date = reader.GetDateTime(5).ToString();
                string manager = reader.GetString(6);
                Sales.Add(ID.ToString() + " ; " + buyer + " ; " + drone + " ;
" + amount.ToString() + " ; " + price.ToString() + " ; " + date + " ; " + manager);
            }
        }
    }
}

```

```

    }
    }
    else Sales.Add("No items");
}
conn.Close();
return Sales.ToArray();
}

```

Б.3.10 Функція GetDrones

```

public string[] GetDrones(NpgsqlConnection conn)
{
    List<string> Drones = new List<string>();
    string sql = "SELECT * FROM drones";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string name = reader.GetString(1);
                int price = Convert.ToInt32(reader.GetValue(2));
                int amount = Convert.ToInt32(reader.GetValue(3));
                string specialist = reader.GetString(4);

                Drones.Add(ID.ToString() + " ; " + name + " ; " +
price.ToString() + " ; " + amount.ToString() + " ; " + specialist);
            }
        }
        else Drones.Add("No items");
    }
    conn.Close();
    return Drones.ToArray();
}

```

Б.3.11 Функція GetManagers

```

public string[] GetManagers(NpgsqlConnection conn)
{
    List<string> Managers = new List<string>();
    string sql = "SELECT * FROM managers";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string login = reader.GetString(1);
                int sales_amount = Convert.ToInt32(reader.GetValue(2));

                Managers.Add(ID.ToString() + " ; " + login + " ; " +
sales_amount.ToString());
            }
        }
        else Managers.Add("No items");
    }
    conn.Close();
    return Managers.ToArray();
}

```

Б.3.12 Функція GetSpecialists

```

public string[] GetSpecialists(NpgsqlConnection conn)
{
    List<string> Specialists = new List<string>();
    string sql = "SELECT * FROM specialists";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();

```

```

conn.Open();
using (DbDataReader reader = cmd.ExecuteReader())
{
    if (reader.HasRows)    //Якщо є рядки
    {
        while (reader.Read())
        {
            int ID = Convert.ToInt32(reader.GetValue(0));
            string login = reader.GetString(1);
            int manufactured_amount =
Convert.ToInt32(reader.GetValue(2));

            Specialists.Add(ID.ToString() + " ; " + login + " ; " +
manufactured_amount.ToString());
        }
    }
    else Specialists.Add("No items");
}
conn.Close();
return Specialists.ToArray();
}

```

Б.3.13 Функція GetSuppliers

```

public string[] GetSuppliers(NpgsqlConnection conn)
{
    List<string> Suppliers = new List<string>();
    string sql = "SELECT * FROM suppliers";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string name = reader.GetString(1);
                string address = reader.GetString(2);
            }
        }
    }
}

```

```

        int cost_of_orders = Convert.ToInt32(reader.GetValue(3));

        Suppliers.Add(ID.ToString() + " ; " + name + " ; " + address
+ " ; " + cost_of_orders.ToString());
    }
}
else Suppliers.Add("No items");
}
conn.Close();
return Suppliers.ToArray();
}

```

Б.3.14 Функція UpdateSpecialist

```

public void UpdateSpecialist(NpgsqlConnection conn, string specialist)
{
    string sql = "UPDATE specialists SET manufactured_amount =
manufactured_amount + 1 WHERE login='" + specialist + "'";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
}

```

Б.3.15 Функція UpdateManager

```

public void UpdateManager(NpgsqlConnection conn, string manager)
{
    string sql = "UPDATE managers SET sales_amount = sales_amount
+ 1 WHERE login='" + manager + "'";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }

}

```

Б.3.16 Функція UpdateDrones

```

        public void UpdateDrones(NpgsqlConnection conn, string name,int
amount)
    {
        string sql = "UPDATE drones SET amount = amount +" + amount + "
WHERE name =" + name + """;
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
    }

```

Б.3.17 Функція UpdateMinusDrones

```

        public void UpdateMinusDrones(NpgsqlConnection conn, string name,
int amount)
    {
        string sql = "UPDATE drones SET amount = amount -" + amount +
" WHERE name =" + name + """;
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
    }

```

Б.3.18 Функція UpdateMaterials

```

        public void UpdateMaterials(NpgsqlConnection conn, string name,int
amount,int price)
    {
        string sql = "UPDATE materials SET in_storage = in_storage +" +
amount+" WHERE name =" + name + """;
        NpgsqlCommand cmd = new NpgsqlCommand();

```

```

cmd.Connection = conn;
cmd.CommandText = sql;
conn.Close();
conn.Open();
using (DbDataReader reader = cmd.ExecuteReader()) { }

sql = "UPDATE materials SET total_price = total_price +" + price +
" WHERE name =" + name + """;
cmd = new NpgsqlCommand();
cmd.Connection = conn;
cmd.CommandText = sql;
conn.Close();
conn.Open();
using (DbDataReader reader = cmd.ExecuteReader()) { }
}

```

Б.3.19 Функція UpdateSupplier

```

public void UpdateSupplier(NpgsqlConnection conn, string name, int
money)
{
    string sql = "UPDATE suppliers SET cost_of_orders =
cost_of_orders +" + money + " WHERE name =" + name + """;
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
}

```

Б.3.20 Функція NewMaterial

```

public string[] NewMaterial(NpgsqlConnection conn, string name)
{
    string sql = "INSERT INTO materials (name,in_storage,total_price)
values (" + name + "" + "," + 0 + "," + 0 + ")";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
}

```

```

        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
        return GetMaterials(conn);
    }

```

Б.3.21 Функція NewBasicComponent

```

    public string[] NewBasicComponent(NpgsqlConnection conn, string
name, string material, int amount_material, int price)
    {
        string sql = "INSERT INTO basic_components
(name,material,amount_of_material,price) values ('" + name + "'" + "," + material +
"," + amount_material + ","+price+ ")";
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
        return GetBasicComponents(conn);
    }

```

Б.3.22 Функція NewComponent

```

    public string[] NewComponent(NpgsqlConnection conn, string name,
int price, string specialist)
    {
        string sql = "INSERT INTO components (name,price,specialist)
values ('" + name + "'" + "," + price + "," + specialist + ")";
        NpgsqlCommand cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
        UpdateSpecialist(conn, specialist);
        return GetComponents(conn);
    }

```

Б.3.23 Функція NewComponentComposition

```

public string[] NewComponentComposition(NpgsqlConnection conn,
string component, string basic, int price)
{
    string sql = "INSERT INTO component_composition
(component,basic_component,price) values ('" + component + "'" + "," + basic + "','"
+ price+ ")";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
    return GetComponentComposition(conn);
}

```

Б.3.24 Функція NewSupplier

```

public string[] NewSupplier(NpgsqlConnection conn, string name,
string address)
{
    string sql = "INSERT INTO suppliers (name,address,cost_of_orders)
values ('" + name + "'" + "," + address + "','" + 0 + ")";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
    return GetSuppliers(conn);
}

```

Б.3.25 Функція NewBuyer

```

public string[] NewBuyer(NpgsqlConnection conn, string name,
string address)
{
    string sql = "INSERT INTO buyers (name,address) values ('" + name
+ "'" + "," + address + ")";
}

```

```

NpgsqlCommand cmd = new NpgsqlCommand();
cmd.Connection = conn;
cmd.CommandText = sql;
conn.Close();
conn.Open();
using (DbDataReader reader = cmd.ExecuteReader()) { }
return GetBuyers(conn);
}

```

Б.3.26 Функція NewGoodOfSupplier

```

public string[] NewGoodOfSupplier(NpgsqlConnection conn, string
supplier, string material, int price)
{
    string sql = "INSERT INTO good_of_suppliers
(supplier,material,price) values ('" + supplier + "'" + "," + material + "','" + price +
"");

    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
    return GetGoodOfSuppliers(conn);
}

```

Б.3.27 Функція NewDelivery

```

public string[] NewDelivery(NpgsqlConnection conn, string supplier,
string material, int price,int amount)
{
    string sql = "INSERT INTO deliveries_made
(supplier,material,price,amount) values ('" + supplier + "'" + "," + material + "','" +
price + "','" + amount + "');

    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
    UpdateSupplier(conn, supplier, price);
}

```

```

        UpdateMaterials(conn, material, amount, price);
        return GetDeliveriesMade(conn);
    }

```

Б.3.28 Функція NewDrone

```

        public string[] NewDrone(NpgsqlConnection conn, string name, int
price, string specialist)
        {
            string sql = "INSERT INTO drones (name,price,amount,specialist)
values ('" + name + "'" + "," + price + "," + 0 + "," + specialist + "')";
            NpgsqlCommand cmd = new NpgsqlCommand();
            cmd.Connection = conn;
            cmd.CommandText = sql;
            conn.Close();
            conn.Open();
            using (DbDataReader reader = cmd.ExecuteReader()) { }
            UpdateSpecialist(conn, specialist);
            return GetDrones(conn);
        }

```

Б.3.29 Функція NewDroneComposition

```

        public string[] NewDroneComposition(NpgsqlConnection conn, string
drone, string component, int price)
        {
            string sql = "INSERT INTO drone_composition
(drone,component,price) values ('" + drone + "'" + "," + component + "," + price +
")";
            NpgsqlCommand cmd = new NpgsqlCommand();
            cmd.Connection = conn;
            cmd.CommandText = sql;
            conn.Close();
            conn.Open();
            using (DbDataReader reader = cmd.ExecuteReader()) { }

            return GetDroneComposition(conn);
        }

```

Б.3.30 Функція NewSale

```

public string[] NewSale(NpgsqlConnection conn, string buyer, string
drone, int amount, int price, string date, string manager)
{
    string sql = "INSERT INTO sales
(buyer,purchased_drone,amount,price,date,manager) values ('" + buyer + "'" + "," +
drone + "','" + amount + "','" + price + "','" + date + "','" + manager + "')";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader()) { }
    UpdateManager(conn, manager);
    return GetSales(conn);
}

```

Б.3.31 Функція GetUser

```

public string GetUser(NpgsqlConnection conn, int id)
{
    string sql = "SELECT ID, login, password, post FROM users
WHERE ID=" + id.ToString();
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    string tmp = null;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows) //Якщо є рядки
        {
            while (reader.Read())
            {
                int ID = Convert.ToInt32(reader.GetValue(0));
                string Name = reader.GetString(1);
                string Pass = reader.GetString(2);
                string Post = reader.GetString(3);
                tmp = "Name:" + Name + "Post:" + Post;
            }
        }
    }
}

```

```

    }

    }
    return tmp;
}

```

Б.3.32 Функція Autor

```

public int Autor(NpgsqlConnection conn, string name, string pass)
{
    string sql = "SELECT ID, login, password, post FROM users
WHERE login='" + name + "'";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    string tmp = null;
    int index = 0, ID = 0;
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                ID = Convert.ToInt32(reader.GetValue(0));
                string Pass = reader.GetString(2);
                if (Pass == pass) { index = 1; }
            }
        }
        else { Console.WriteLine("Wrong username"); return 0; }
    }

    if (index == 1) tmp = GetUser(conn, ID);
    else { Console.WriteLine("Wrong Password"); return 0; }
    if (tmp.Contains("Storage") == true) return 1;
    else if (tmp.Contains("specialist") == true) return 2;
    else return 3;
}

```

Б.3.33 Функція UsernameChecker

```

public int UsernameChecker(NpgsqlConnection conn, string name)
{
    string sql = "SELECT ID FROM users WHERE login=" + name
+ """;
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
    conn.Open();
    using (DbDataReader reader = cmd.ExecuteReader())
    {
        if (reader.HasRows)    //Якщо є рядки
        {
            while (reader.Read())
            {
                if (!reader.IsDBNull(0)) return 1;
            }
        }
    }

    return 0;
}

```

Б.3.34 Функція NewUser

```

public string NewUser(NpgsqlConnection conn, string name, string
pass, string post)
{
    if (UsernameChecker(conn, name) == 1) return "Name already
used";

    string sql = "INSERT INTO users (login,password,post) values ("
+ name + "" + ","
+ "" + pass + "" + "," + "" + post + "" + ")";
    NpgsqlCommand cmd = new NpgsqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sql;
    conn.Close();
}

```

```

        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
        if (post.Contains("specialist")) sql = "INSERT INTO specialists
(login,manufactured_amount) values ('" + name + "'" + "," + "0" + ")";
        else if (post.Contains("seller")) sql = "INSERT INTO managers
(login,sales_amount) values ('" + name + "'" + "," + "0" + ")";
        else sql = "Select * From users";
        cmd = new NpgsqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        conn.Close();
        conn.Open();
        using (DbDataReader reader = cmd.ExecuteReader()) { }
        if (post.Contains("storage") == true) return "1";
        else if (post.Contains("specialist") == true) return "2";
        return "3";
    }

```

Б.3.35 Функція GetDBConnection (string host, int port, string database, string username, string password)

```

    public static NpgsqlConnection
    GetDBConnection(string host, int port, string database, string
username, string password)
    {
        // Connection String.
        String connString = "Server=" + host + ";Database=" + database
        + ";port=" + port + ";User Id=" + username + ";password=" +
password;
        NpgsqlConnection conn = new NpgsqlConnection(connString);
        return conn;
    }

```

Б.3.36 Функція GetDBConnection

```

    public static NpgsqlConnection GetDBConnection()
    { //DataBase connect information
        string host = "localhost";
        int port = 5432;
        string database = "Drone_production";
        string username = "postgres";
    }

```

```
        string password = "admin";  
        return GetDBConnection(host, port, database, username, password);  
    }  
}
```

ДОДАТОК В
Слайди презентації

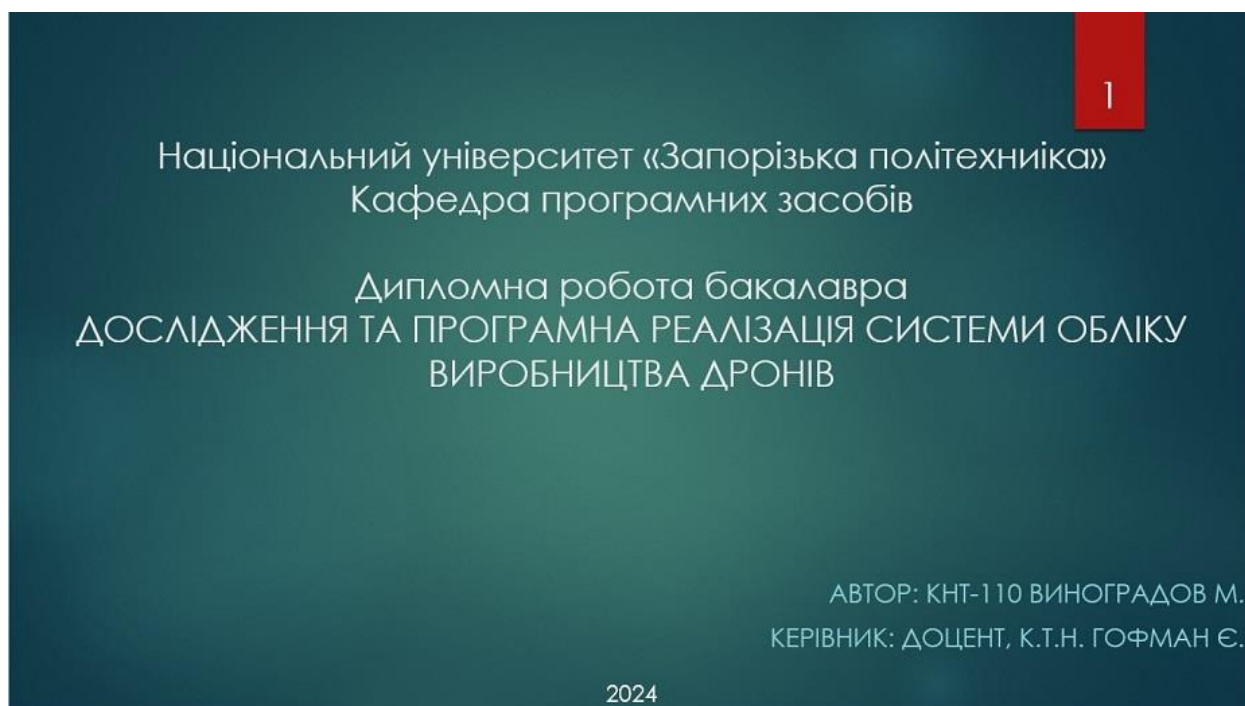


Рисунок Г.1 – Слайд №1

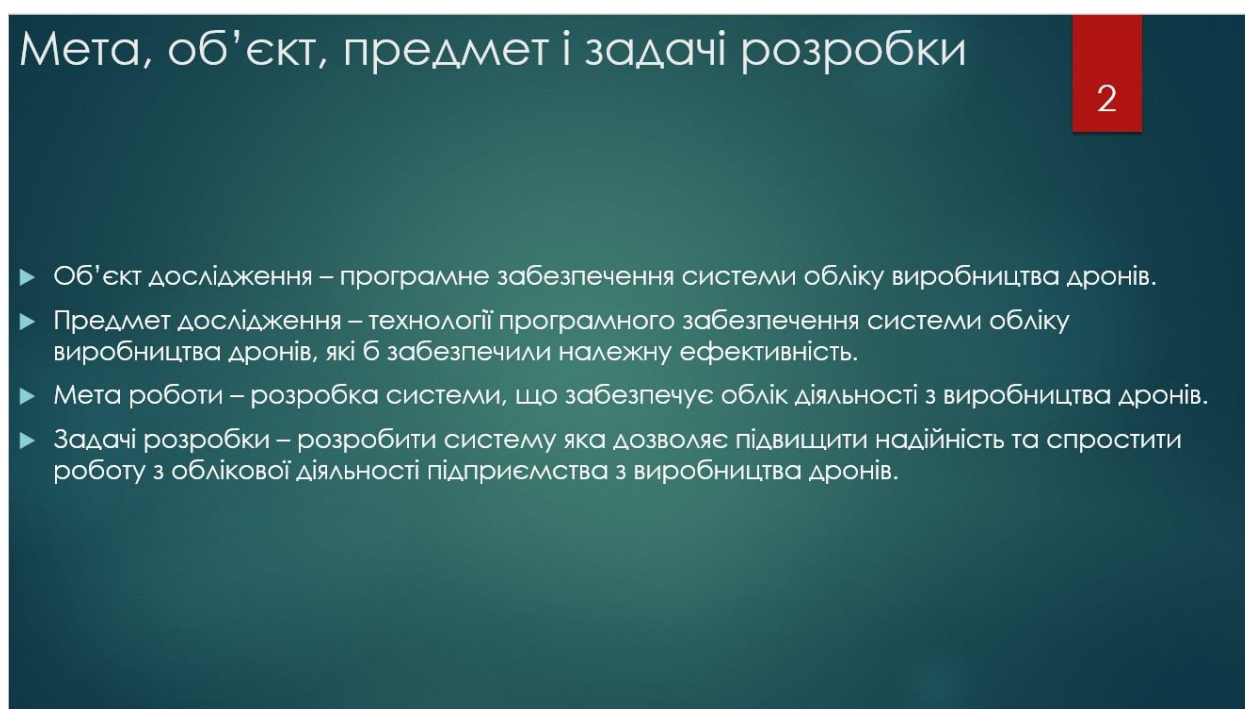


Рисунок Г.2 – Слайд №2

Огляд існуючих аналогів

3

	HugeProfit	Торгсофт	УкрСклад	Власна розробка
Можливість роботи через Internet	+	+	-	+
Можливість роботи через локальну мережу	-	+	+	+
Зручний інтерфейс	+	+	-	+
Формування звітів	-	-	+	+
Підтримка різних типів користувачів	+	-	-	+

Рисунок Г.3 – Слайд №3

Структурна схема програми

4

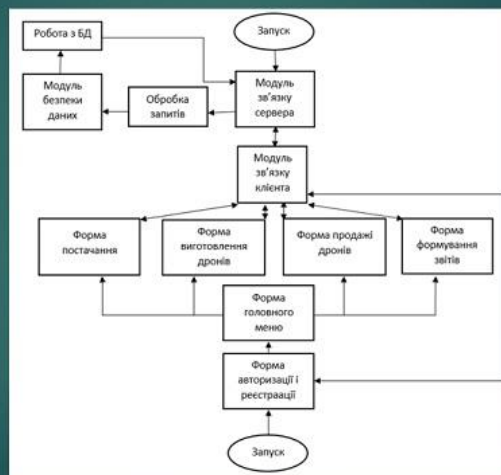


Рисунок Г.4 – Слайд №4

Схема функціонування розробленої програми

5

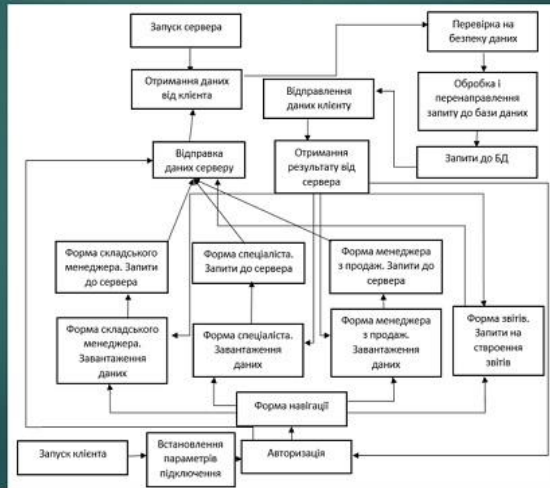


Рисунок Г.5 – Слайд №5

Схема бази даних

6

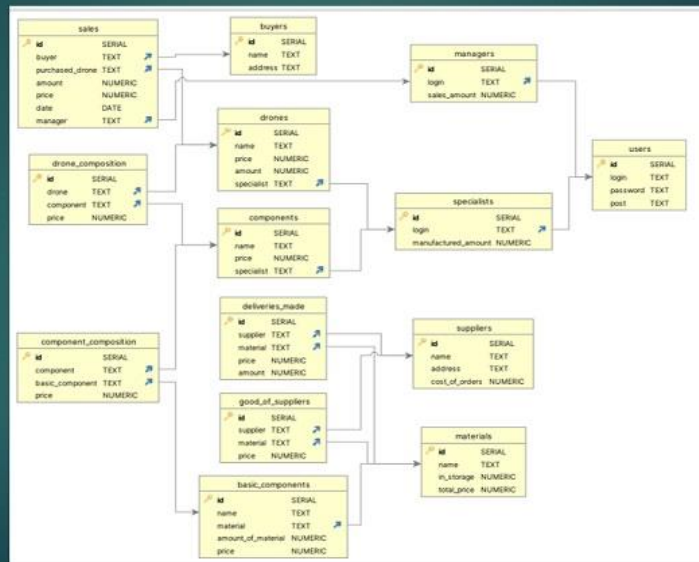


Рисунок Г.6 – Слайд №6

Обґрунтування вибору засобів реалізації

7

	C++	C#	Python	Java	Характеристика	Microsoft Visual Studio 2017 Community Edition	JetBrains Rider	MonoDevelop
Підтримка .NET	+	+	-	-	Багатоплатформеність	-	+	-
Многопотоковість	+	+	+-	+	Швидкість модернізації	+	-	-
Автоматичне керування пам'яттю	+	-	+	+	Наявність візуального конструктора	+	+	+
Робота з БД	+	+	+-	+	Підтримка репозиторіїв	+	+	+
Типізація	+	+	+-	+	Доступність	+	+	-
					Підтримка .Net	+	+	+-

Рисунок Г.7 – Слайд №7

Графічні форми клієнта

8

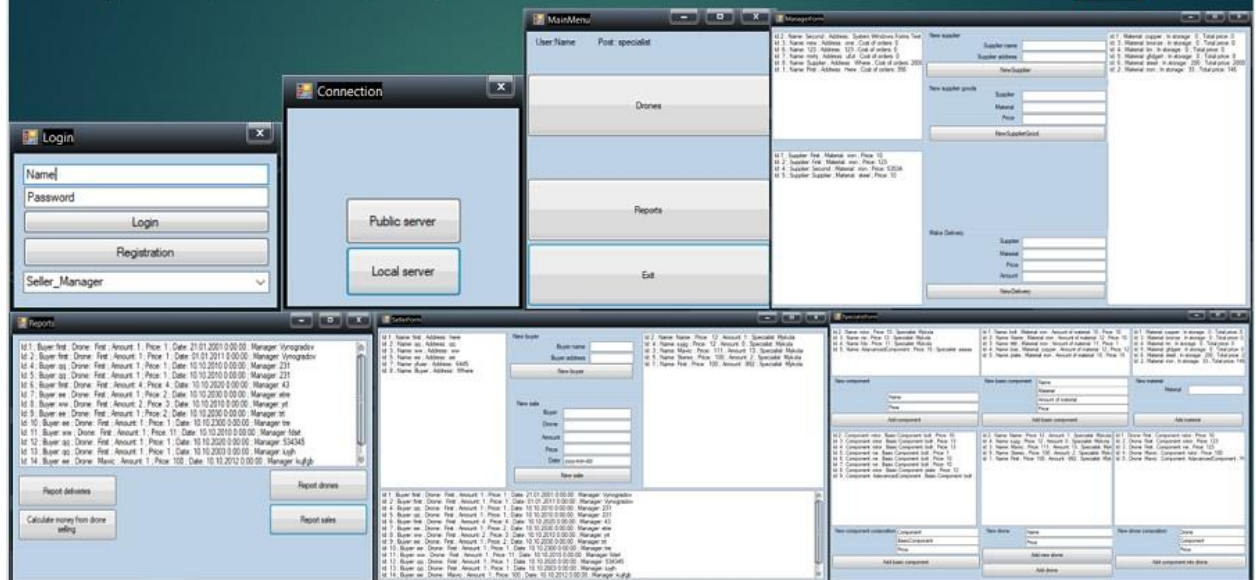


Рисунок Г.8 – Слайд №8

Серверна частина програми

9

```

D:\Drone_production\Drone_ServerApp\Drone_ServerApp\bin\Debug\Drone_ServerApp.exe
-----
Client connect.
From client: GetGoodOfSuppliers
Send to client: 1 ; First ; iron ; 10 ; 2 ; First ; iron ; 123 ; 4 ; Second ; iron ; 53534 ; 5 ; Supplier ; steel ; 10 ;

Client connect.
From client: Registration: ?????? Password????? storage_manager
Send to client: 1

Client connect.
From client: GetDeliveriesMade
Send to client: 1 ; First ; iron ; 10 ; 12 ; 2 ; First ; iron ; 123 ; 31 ; 3 ; First ; iron ; 123 ; 100 ; 4 ; Supplier ;
steel ; 10 ; 100 ; 5 ; Supplier ; steel ; 990 ; 100 ; 6 ; Supplier ; steel ; 1000 ; 0 ; 7 ; First ; iron ; 100 ; 10 ;

Client connect.
From client: GetDrones
Send to client: 2 ; Name ; 12 ; 1 ; Mykola ; 4 ; iuyjg ; 12 ; 0 ; Mykola ; 3 ; Mavic ; 111 ; 13 ; Mykola ; 5 ; Stereo ;
100 ; 2 ; Mykola ; 1 ; First ; 100 ; -992 ; Mykola ;

Client connect.
From client: GetSales
Send to client: 1 ; first ; First ; 1 ; 1 ; 21.01.2001 0:00:00 ; Vynogradov ; 2 ; first ; First ; 1 ; 1 ; 01.01.2011 0:0
0:00 ; Vynogradov ; 4 ; qq ; First ; 1 ; 1 ; 10.10.2010 0:00:00 ; 231 ; 5 ; qq ; First ; 1 ; 1 ; 10.10.2010 0:00:00 ; 23
1 ; 6 ; First ; First ; 4 ; 4 ; 10.10.2020 0:00:00 ; 42 ; 7 ; ee ; First ; 1 ; 2 ; 10.10.2020 0:00:00 ; etre ; 8 ; ww ;
First ; 2 ; 3 ; 10.10.2010 0:00:00 ; yrt ; 9 ; ee ; First ; 1 ; 2 ; 10.10.2030 0:00:00 ; trt ; 10 ; ee ; First ; 1 ; 1 ;
10.10.2300 0:00:00 ; tre ; 11 ; ww ; First ; 1 ; 11 ; 10.10.2010 0:00:00 ; fdrst ; 12 ; qq ; First ; 1 ; 1 ; 10.10.2020
0:00:00 ; 534345 ; 13 ; qq ; First ; 1 ; 1 ; 10.10.2003 0:00:00 ; iuyjh ; 14 ; ee ; Mavic ; 1 ; 100 ; 10.10.2012 0:00:0
0 ; kujfjb ; 15 ; Buyer ; First ; 1000 ; 12 ; 10.10.2010 0:00:00 ; jgjh ;
  
```

Рисунок Г.9 – Слайд №9

Висновки

10

- Розроблено програмну систему, яка призначена для обліку виробництва дронів.
- Розроблена система дозволяє спростити та підвищити надійність обліку виробництва дронів.
- Галузь використання – підприємство, що займається виготовленням дронів.

Рисунок Г.10 – Слайд №10