**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ**

**Кафедра комп’ютерних наук та прикладної математики**

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

на тему

«Розробка бази даних Записна книга приватного підприємця»

Виконала студентка ННІ

АКОТ: КН-31

Федун Назарій Олегович

Керівник:

 кандидат економічних наук,

доцент кафедри комп'ютерних наук

та прикладної математики,

Бачишина Л. Д.

Рівне – 2022

**ЗМІСТ**

1. ВСТУП…………………………………………………………………….. 3
2. ТЕОРЕТИЧНА ЧАСТИНА………………………………………………. 5
   1. Технології використання………………………………………….. 5
   2. Опис таблиць БД…………………………………………………… 5
   3. Опис проектів………………………………………………………. 8
   4. Встановлення пакетів……………………………………………… 8
3. ПРАКТИЧНА ЧАСТИНА (КОНСОЛЬНИЙ ПРОЕКТ)………………... 9
4. ПРАКТИЧНА ЧАСТИНА (ДЕСКТОПНИЙ ПРОЕКТ)……………….. 22
5. ВИСНОВОК……………………………………………………………… 39
6. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ…………………………... 40
7. **ВСТУП**

**Тема дослідження:** Розробка бази даних «Записна книга приватного підприємця».

**Актуальність проблеми.** З розвитком суспільства зростає кількість накопичених та здобутих нею знань, які потребують коректної систематизації, яка згодом спростила б роботу з нею. Але матеріальні носії інформації, такі як камінь, глина, папір і т.д. вичерпали себе або повністю, або частково, адже є недовговічнТехнологіиим, схильними до руйнування через численні зовнішні чинники. Тому з часом людина вигадала структури, які призначені саме для зберігання, зміни та обробки взаємозалежної інформації переважно великих обсягів – базу даних (БД).

В наші дні її використовують переважно в усьому цифровому світі: починаючи з малих і продовжуючи великими ІТ технологіями та винаходами. На сьогодні, майже неможливо уявити якийсь сайт з продажу товарів або соціальну мережу без БД, адже є необхідність зберігати логіни, пошти, паролі, фотографії, пости користувачів та ін.

Модернізація інформаційної сфери так само торкнулася і бізнес-середовища, де є надгостра потреба у чіткій та систематизованій роботі з даними. Зокрема, це стосується записних книжок фізичних осіб підприємців (ФОП). Адже опрацювання, зміна та зберігання таких документів потребує застосування додаткових ресурсів при наявних недоліках у системі зберігання, заповнення та т.і.. А з часом все більше і більше осіб цікавляться реалізацією власної справи, що збільшує кількість запитів у цій сфері.

**Мета:** розробка програми записної книжки приватного підприємця з підключеннями бази даних.

Серед **завдань курсової роботи** можна виокремити наступні пункти:

* Проаналізувати сьогоденні умови ведення записних книжок підприємцями
* Виокремити недоліки сучасної паперової системи ведення записів ФОП
* Визначити ставлення самих підприємців до паперового ведення записів ФОП
* Дослідити сучасне ведення записів ФОП підприємцями
* Створити комп’ютерну програму яка значно полегшує ведення записів ФОП

**Об’єктом** курсової роботи виступає база даних «Записна книга приватного підприємця».

**Суб’єктом** є дослідження сучасного стану ведення записів приватними підприємцями та реалізація бази даних, яка потенційно полегшить роботу з масивами.

Для реалізації проекту бази даних було використано такі **методи дослідження**: спостереження за паперовим веденням записів ФОП, аналіз спостереження та самостійне тестування записників

**Коротка характеристика структури виконаного дослідження:**

1. Аналіз матеріалів
2. Виявлення недоліків
3. Визначення ставлення ФОП до сучасного ведення записів
4. Дослідження ведення записів ФОП
5. Створення програми для ПК, яка значно полегшить запис для ФОП

1. **ТЕОРЕТИЧНА ЧАСТИНА**
   1. **Технології використання**

Співвіднісши цілі та завдання дослідження з сучасними реаліями та можливостями, було обрано найбільш доцільну програму для написання коду та оформлення візуального виду програми, а саме «**Visual Studio 2022**». Робота з базою даних та створення таблиць буде здійснена через програму «**Microsoft SQL Server Management Studio**».

* 1. **Опис таблиць БД**

Таблиця «**tblManufacturers**» (Виробники)

Поле «**Id**» - ключове

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип поля | Пояснення |
| **Id** | **INTEGER** | **Унікальний ідентифікатор Виробника** |
| Name | NVARCHAR(100) | Назва компанії Виробника |

Таблиця «**tblGroceries**» (Продукти)

Поле «**Id**» - ключове

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип поля | Пояснення |
| **Id** | **INTEGER** | **Унікальний ідентифікатор**  **Продукту** |
| Name | NVARCHAR(100) | Назва продукту |

Таблиця «**tblStorage**» (Склад)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип поля | Пояснення |
| **ManufacturerId** | **INTEGER** | **Унікальний ідентифікатор Виробника** |
| **ProductId** | **INTEGER** | **Унікальний ідентифікатор Продукту** |
| Price | INTEGER | Ціна продукту |
| Count | INTEGER | Кількість продуктів |

Таблиця «**tblCustomers**» (Клієнти)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип поля | Пояснення |
| **Id** | **INTEGER** | **Унікальний ідентифікатор Клієнта** |
| Name | NVARCHAR(100) | Ім’я клієнта |
| Phone | NVARCHAR(50) | Номер клієнта |
| PurchaseSum | INTEGER | Загальна сума покупки за весь час |

Таблиця «**tblNotebook**» (Записна Книга)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип поля | Пояснення |
| **Id** | **INTEGER** | **Унікальний ідентифікатор Замовлення** |
| Date | DATE | Дата покупки |
| ProductId | INTEGER | Ідентифікатор Продукту |
| ManufacturerId | INTEGER | Ідентифікатор Виробника |
| Count | INTEGER | Кількість Продуктів |
| TotalSum | INTEGER | Сума до сплати |
| CustomerId | INTEGER | Ідентифікатор Клієнта |
| Discount | INTEGER | Знижка |

Нормалізувавши БД, отримаємо дану схему зв’язків між створенними таблицями (дизайн таблиць створений в програмі Microsoft SQL Server Management):

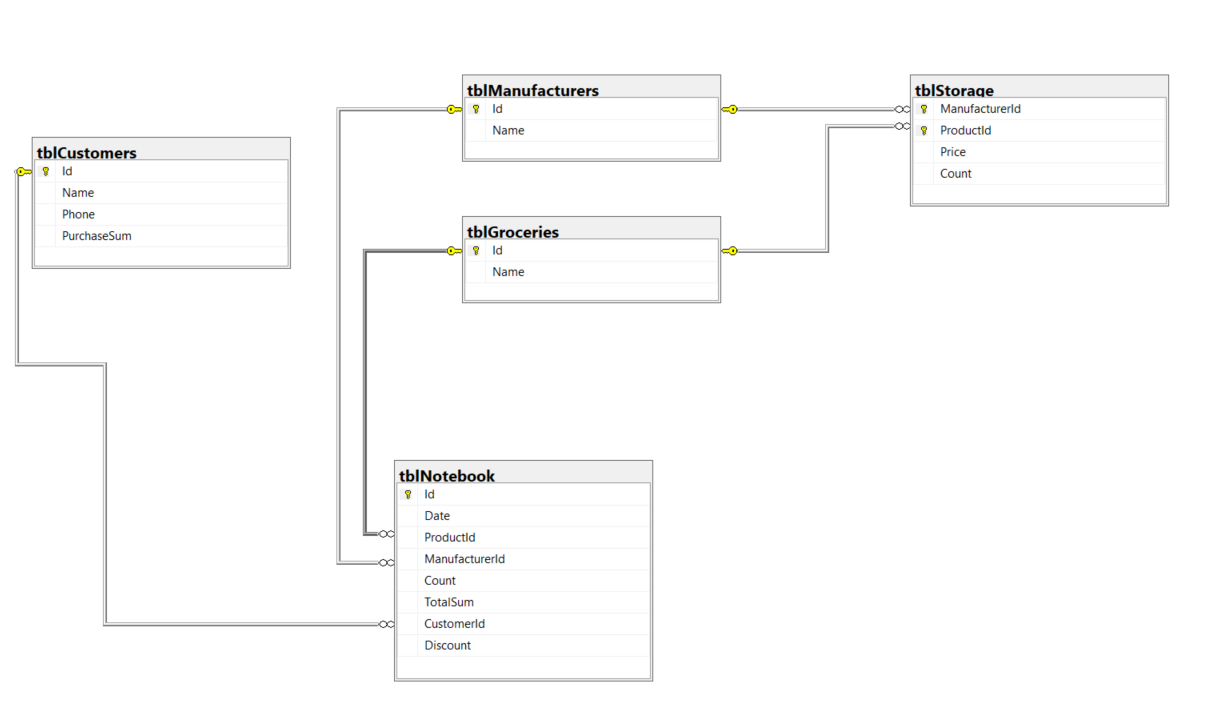


Рис. 1. Схема зв’язків між таблицями

**Умови збереження цілісності даних**

* В таблиці «**tblManufacturers**» знаходяться назви компаній з якими співпрацює ФОП
* В таблиці «**tblGroceries**» знаходяться назви продуктів які ФОП збирається продавати
* В таблиці «**tblStorage**» знаходиться вся інформація про товари, що є в наявності. Ця таблиця має зв’язок з двома таблицями: «**tblManufacturers**», «**tblGroceries**». Зв’язок з таблицею де знаходяться Виробники встановлено по полю «**ManufacturerId**». Зв’язок з таблицею де знаходяться Продукти встановлено по полю «**tblGroceries**».
* В таблиці «**tblCustomers**» знаходиться певна інформація про клієнтів
* В таблиці «**tblNotebook**» знаходиться вся інформація, що потрібна для записної книги ФОП. Ця таблиця має зв’язок з трьома таблицями: «**tblManufacturers**», «**tblGroceries**», «**tblCustomers**». Зв’язок з таблицею де знаходяться Виробники встановлено по полю «**ManufacturerId**». Зв’язок з таблицею де знаходяться Продукти встановлено по полю «**ProductId**». Зв’язок з таблицею де знаходяться Клієнти встановлено по полю «**tblCustomers**»
  1. **Опис проектів**

Для початку роботи необхідно створити проект для подальшої коректної реалізації програми. Виходячи з поставлених завдань та цілей курсової роботи було вирішено обрати два типи проектів, а саме: **консольний**, **десктопний**. В консольному проекті при компіляції буде створюватись БД та таблиці. Після створення таблиць, деякі з них будуть автоматично заповнюватись даними. В десктопному проекті, вже буде намальована сама програма і під’єднана до БД. Весь код буде написаний на мові **C#**.

* 1. **Встановлення пакетів**

Для роботи з БД підключимо пакет «**System.Data.SqlClient**». Необхідно створити **json** файл з назвою «**appsettings.json**» в якому будемо зберігати рядок, що відповідає за підключення до локального хоста в якому знаходяться всі БД на ПК.

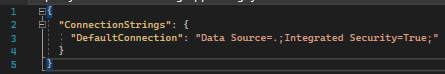


Рис. 2. Файл appsettings.json

Для того, щоб в подальшому отримувати доступ до цього файлу. Потрібно завантажити пакет «**Microsoft.Extensions.Configuration.Json**».

Встановимо пакет «**Bogus**». Це простий генератор різноманітних фейкових даних.

1. **ПРАКТИЧНА ЧАСТИНА (КОНСОЛЬНИЙ ПРОЕКТ)**

Для початку потрібно підключитись до локального хоста, де знаходяться БД. Створимо два поля: **con** і **cmd**. **con** – буде відповідати за підключення до БД. **cmd –** буде відповідати за створення sql команд та їх виконання



Рис. 3. Поля для роботи з БД

Створимо поле, в якому будемо мати ім’я БД, яку хочемо створити. Назвемо його **DbName.**

****

Рис. 4. Поле з назвою БД

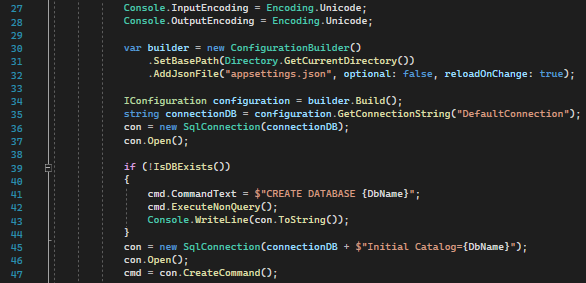


Рис. 5. Підключення до БД

На Рис. 5. З 27 по 37 рядки виконується підключення до локального хосту. В 39 рядку виконується перевірка в методі «**IsDBExists**»чи існує вже БД з такою назвою, яка записана в полі **DbName**. Якщо такої бази даних поки не існує, то через sql команду створюємо її. В 45-47 рядках підключаємось до створеної БД і підключаємо до неї cmd, щоб sql команди виконувались в ній.

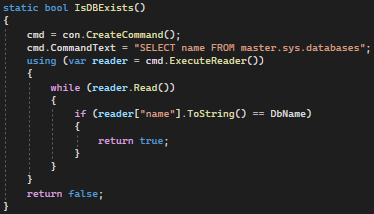


Рис. 6. Метод перевірки існування БД

Далі йде створення таблиць в БД. Створюватись вони будуть через sql команди. Створимо папку «**SqlTables**» в якій будуть зберігатись файли з цими командами.

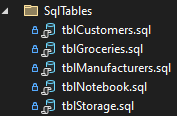


Рис. 7. Вміст папки SqlTables

В кожному файлі буде виконуватись перевірка чи існує таблиця вже таким іменем.



Рис. 8. Перевірка на існування таблиці в БД

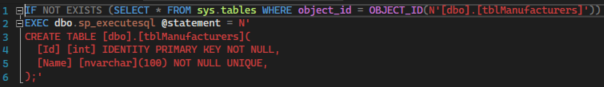


Рис. 9. Створення таблиці **tblManufacturers**

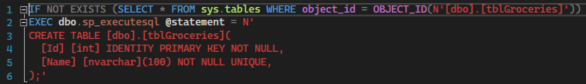
****

Рис. 10. Створення таблиці **tblGroceries**

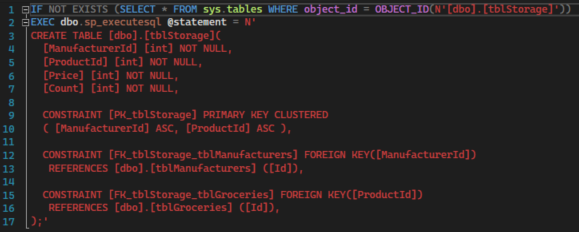
****

Рис. 11. Створення таблиці **tblStorage**

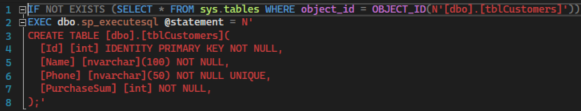


Рис. 12. Створення таблиці **tblCustomers**

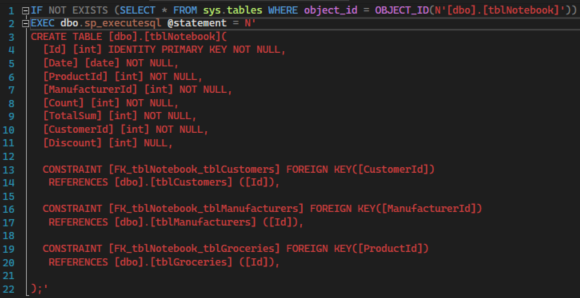
****

Рис. 13. Створення таблиці **tblNotebook**

Створимо поле в якому буде зберігатись ім’я дерикторії в якій містяться файли з sql командами по створенню таблиць. Назвемо його **dirSql.**

****

Рис. 14. Поле з назвою директорії sql команд

Створимо метод «**ReadSqlFile»** який буде отримувати назву файлу, зчитувати його і повертати.

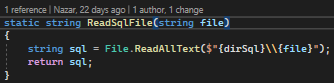


Рис. 15. Метод ReadSqlFile

Створимо метод «**ExecuteCommandFromFile**», який буде отримувати назву файлу, отримувати його вміст використовуючи метод «**ReadSqlFile**» та викликати цю sql команду через **cmd**.

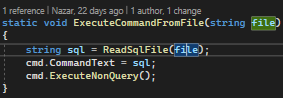


Рис. 16. Метод ExecuteCommandFromFile

Створимо метод «**GenerateTables**». Створимо масив string в якому будуть зберігатись назви наших файлів, та будем через масив foreach викликати їх через метод **ExecuteCommandFromFile**.

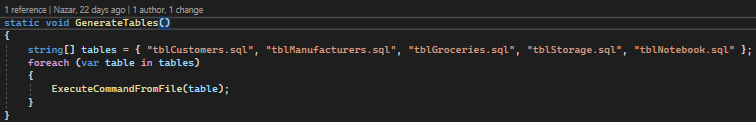


Рис. 17. Метод GenerateTables

Після виклику методу «**GenerateTables**», всі таблиці будуть створені в БД.

Заповнимо таблицю «**tblManufacturers**». Для автоматичного заповнення створимо json файл «**Manufacturers.json**», в якому будуть зберігатись десять компаній, обрані у рандомному порядку. Файл буде зберігатись в директорії «**JSON\_Objects**».



Рис. 18. Вміст папки JSON\_Objects

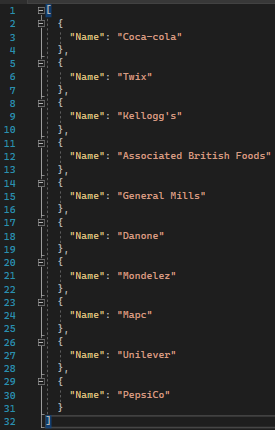


Рис. 19. Вміст файлу Manufacturers.json

Створимо клас «**Manufacturer**» в якому буде пропс, що буде зберігати ім’я виробника. Він буде зберігатись в Директорії «**Models**».

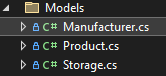


Рис. 20. Вміст папки Models

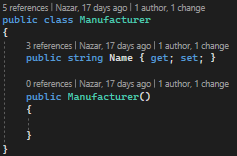


Рис. 21. Клас Manufacturer

Створимо метод «**GenerateManufacturers**». В методі будем зчитувати дані «**Manufacturer.json**», та заповнювати ними саму таблицю «**tblManufacturers**» через **DataTable**. Через нього заповнення таблиці буде виконуватись в 10 разів швидше.

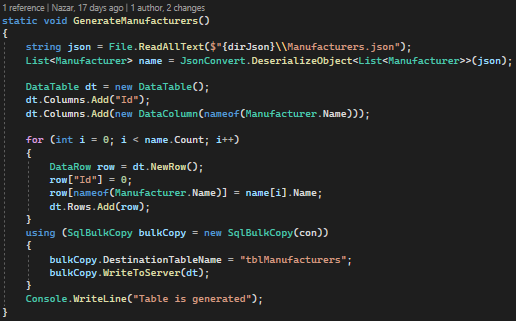


Рис. 22. Метод GenerateManufacturers

Після виклику методу «**GenerateManufacturers**» отримаємо результат в таблиці «**tblManufacturers**» на Рис. 23.

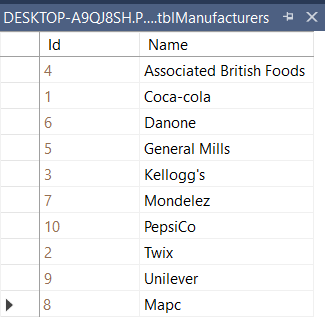


Рис. 23. Вміст таблиці tblManufacturers після виклику GenerateManufacturers

Заповнимо таблицю «**tblGroceries**». Для автоматичного заповнення створимо json файл «**Groceries.json**», в якому будуть зберігатись 14 різноманітних продуктів. Файл буде зберігатись в директорії «**JSON\_Objects**» (Див. Рис. 18).

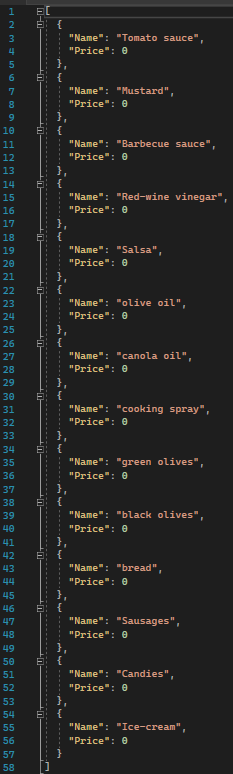


Рис. 24. Вміст файлу Groceries.json

Створимо клас «**Product**», в якому буде пропс, що буде зберігати ім’я продукту. Він буде зберігатись в Директорії «**Models**» (Див. Рис. 20).

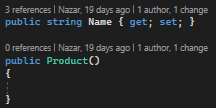


Рис. 25. Клас Product

Створимо метод «**GenerateGroceries**». В методі будем зчитувати дані «**Groceries.json**», та заповнювати ними саму таблицю «**tblGroceries**» через **DataTable**.

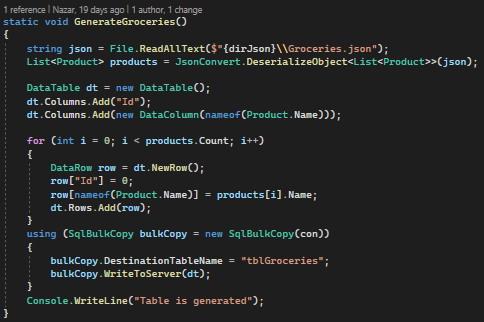


Рис. 26. Метод GenerateGroceries

Після виклику методу «**GenerateGroceries**» отримаємо результат в таблиці «**tblGroceries**», який можна переглянути на Рис. 27.



Рис. 27. Вміст таблиці tblGroceries після виклику GenerateGroceries

Заповнимо таблицю «**tblStorage**». Для початку створимо клас «**Storage**», в якому будуть чотири пропса:

* **ManufacturerId** – містить **унікальний ідентифікатор виробника**
* **ProductId** – містить **унікальний ідентифікатор продукту**
* **Price** – містить ціну продукту за одиницю
* **Count** – містить кількість продуктів

Цей клас буде зберігатись в Директорії «**Models**» (Див. Рис. 20).

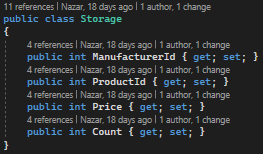


Рис. 28. Клас Storage

Перед тим як створювати метод по заповненню таблиці «**tblStorage**» потрібно створити методи які будуть виводити **унікальні ідентифікатори Продуктів** та **Виробників**.

Створимо метод «**GenerateGroceriesId**». Він буде виводити список **унікальних ідентифікаторів Продуктів**.

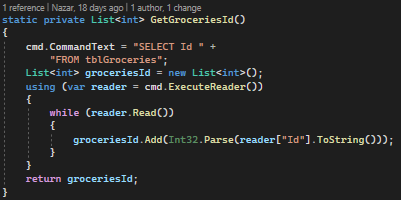


Рис. 29. Метод GenerateGroceriesId

Створимо метод «**GetManufacturersId**». Він буде виводити список **унікальних ідентифікаторів Виробників**.

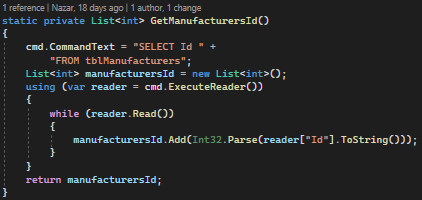


Рис. 30. Метод GenerateManufacturersId

Створимо метод «**GenerateStorage**». Він буде заповнювати таблицю «**tblStorage**» в БД.

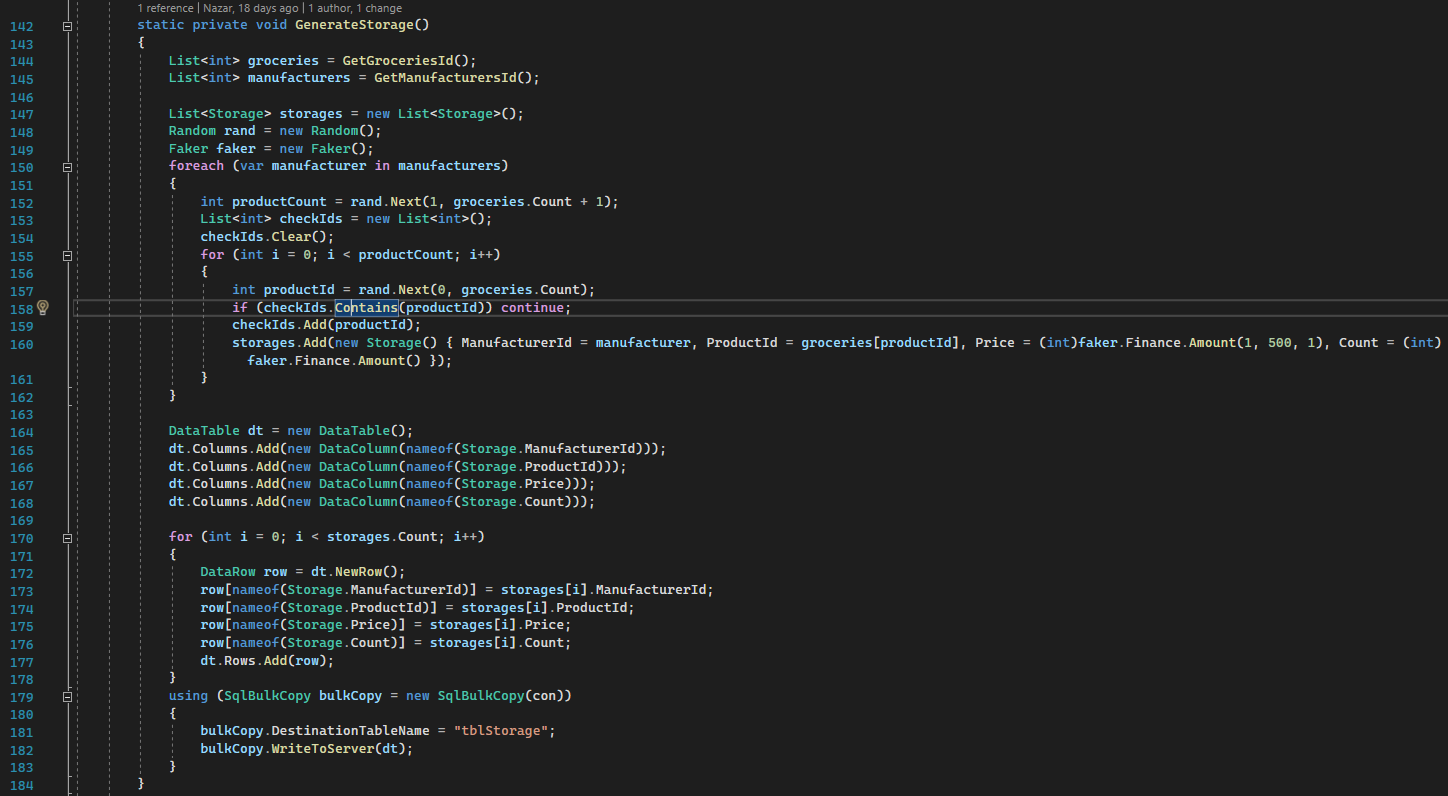


Рис. 31. Метод GenerateStorage

У ряд. 144-145 отримуємо списки **унікальних ідентифікаторів Продуктів** та **Виробників**. У ряд.147 створюємо список, під назвою «**storages**», в якому будуть міститись об’єкти які пізніше запишуться в таблицю «**tblStorage**». У ряд.148 створюємо зміну типу **Random**, яка буде виводити рандомні числа. У ряд.149 створюємо зміну типу **Faker**, під назвою «**faker**», яка нам буде видавати фейкові дані. У ряд.150 створюємо масив, що буде здійснюватись ітерація по **унікальним ідентифікаторам Виробників**. У ряд. 152 отримуємо рандомну одиницю під назвою «**productCount**», яка буде відповідати за кількість видів продуктів які буде мати **Виробник**. У ряд.153-154 створюємо список під назвою «**checkIds**», та далі відразу очищаємо. Цей список в подальшому буде містити **унікальні ідентифікаторі Продуктів**, які вже має поточний **Виробник**. У ряд. 155 створюємо вкладений масив, що буде ітеровуватись «**productCount**»разів. У ряд.157 отримуємо рандомний **унікальний ідентифікатор Продукту** під назвою «**productId**». У ряд.158 перевіряємо чи список «**checkIds**» вже містить «**productId**». Якщо так, то пропускаємо подальшу роботу. Якщо ні, то продовжуємо роботу коду і добавляємо поточний **унікальний ідентифікатор Продукту** в список «**checkIds**» у ряд. 159. У ряд. 160 добавляємо об’єкт в список «**storages**». У змінну «**ManufacturerId**» записуємо поточний **унікальний ідентифікатор Виробника**. У змінну «**ProductId**» записуємо поточний **унікальний ідентифікатор Продукту**. У змінну «**Price**» за допомогою «**faker**» записуємо випадкову число від 1 до 500. У змінну «**Count**» за допомогою «**faker**» записуємо випадкове число по дефолту. Після того, як завершиться масив, що починається у ряд. 150 – йде заповнення таблицю «**tblStorage**» за допомогою **DataTable**. Таким чином, у нас заповниться таблиця без помилок. Після виклику методу «**GenerateStorage**» отримаємо результат в таблиці «**tblStorage**». На рис. 32. Можна переглянути як будуть виглядати дані в таблиці.

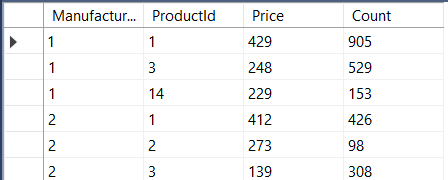


Рис. 32. Вигляд даних в таблиці tblStorage

1. **ПРАКТИЧНА ЧАСТИНА (ДЕСКТОПНИЙ ПРОЕКТ)**

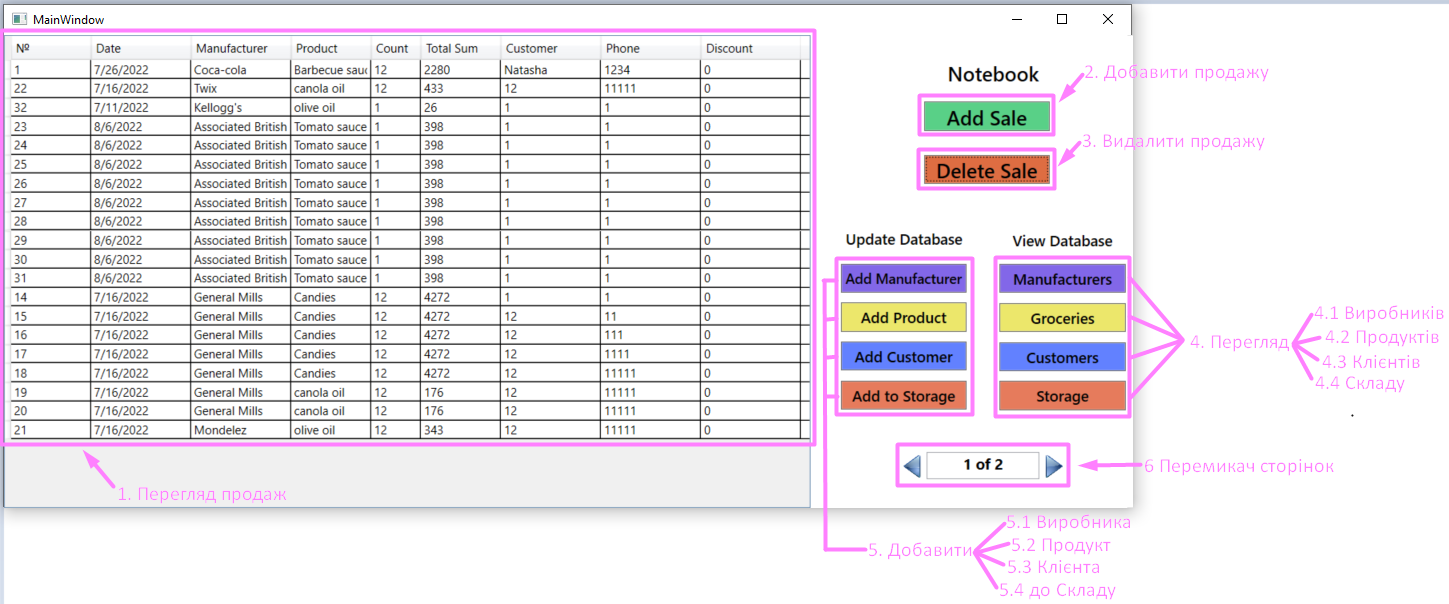
Перейдемо до самого створення програми. Кінцевий вид програми можна переглянути на рис. 33. 

Рис. 33. Вид програми

1. **Перегляд продаж** – відображення таблиці «**tblNotebook**». Відображення відбувається за допомогою вбудованого інструменту DataGrid.
2. **Добавити продажу** – при натиску на цю кнопку відкривається вікно, в якому можна вписати всі потрібні дані для нового продажу.
3. **Видалити продажу** – при натиску на кнопку видаляється вибраний продаж в DataGrid. Кілкість продуктів яка була продана – повертається назад на склад.
4. **Перегляд**: 4.1 **Виробників** – перегляд таблиці «**tblManufacturers**»; 4.2 **Продуктів** – перегляд таблиці «**tblGroceries**»; 4.3 **Клієнтів** – перегляд таблиці «**tblClients**»; 4.4 **Складу** – перегляд таблиці «**tblStorage**».
5. **Додати**: 5.1 **Виробника –** при натиску на кнопку, відкривається вікно в якому можна добавити нового виробника; 5.2 **Продукт** – при натиску на кнопку, відкривається вікно в якому можна добавити новий продукт; 5.3 **Клієнта** – при натиску на кнопку, відкривається вікно в якому можна добавити нового клієна; 5.4 **до Складу** – при натиску на кнопку, відкривається вікно в якому можна додати новий об’єкт до складу.
6. **Перемикач сторінок** – показує на який користувач знаходиться сторінці, та їх максимальну кількість. Також за допомогою трикутників можна перемикатись між сторінками.

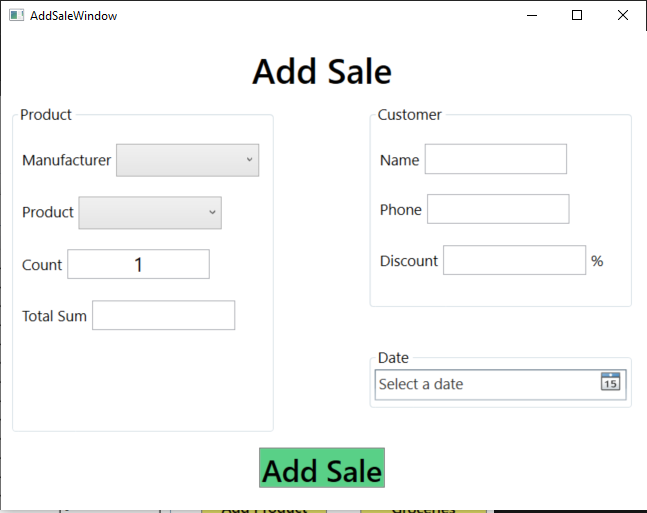
Розглянемо кнопку під номером 1. При натиску на неї відкривається вікно (Див. Рис. 34)

Рис. 34. Вікно по добавці продажу

Вибір **Виробника** та **Продукту** здійснюється за допомогою інструменту «**ComboBox**». При його натиску висвітлюються варіанти які можна обрати. На приклад при натиску на **ComboBox** **Виробника**, висвітляться всі **Виробники** з таблиці «**tblManufacturers**» (Див. Рис. 35).

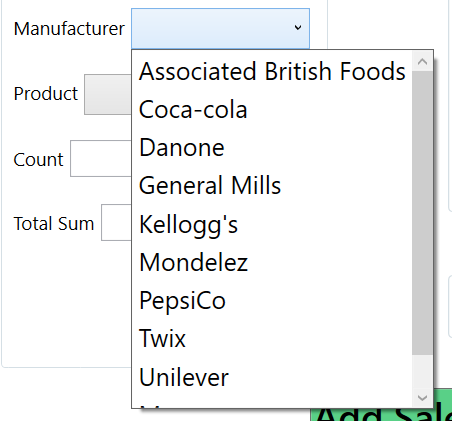
ї

Рис. 35. Вигляд ComboBox Виробника

ComboBox **Продукту** буде пустий поки не буде обрано **Виробника** (Див Рис. 36).

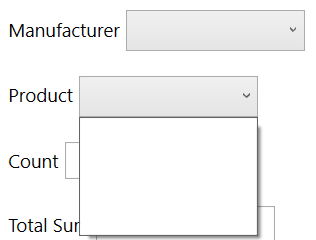


Рис. 36. Початковий вигляд ComboBox Продукту

Після того як був обраний **Виробник** - **ComboBox Продукту** буде містити ті продукти, які є у вибраного **Виробника** (Див. Рис. 37)

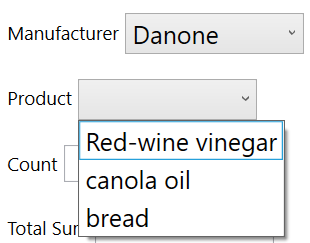


Рис. 37. Вигляд ComboBox Продукту після вибору Виробника

Після вибору **Продукту**, інструмент **TextBox** загальной суми (TotalSum) – буде висвітлювати суму той кількості яка введена в **TextBox** кількості (Count) (Див. Рис. 38).

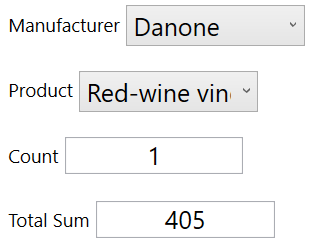


Рис. 38. Вигляд вікна після вибору продукту

У **TextBox** який відповідає за кількість продуктів. Передбачено:

* Ввід лише цифр
* Попередження, що введеної кількості продукту немає в складі (Див. Рис. 39)

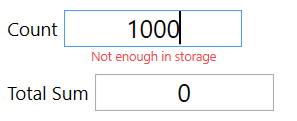


Рис. 39. Помилка про недостачу продукту в складі

У **TextBox** загальной суми передбачена заборона зміни ціни. Тобто можливий лише перегляд цього інструменту

Передбачений перевибір **Виробника** у його **ComboBox**. Якщо **Виробника** перевибирають, то:

* Переініціалізовується **ComboBox Продукту** і на початку нічого не вибрано в ньому
* Кількість рівна одиниці
* Загальна сума рівна нулю

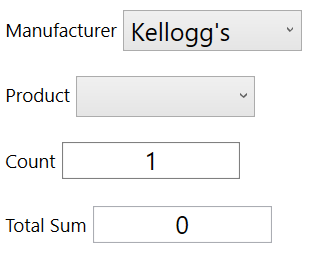


Рис. 40. Вигляд вибору продукту після перевибору Виробника

Далі потрібно вибрати користувача. Можна вводити користувача, який не існує в таблиці «**tblCustomers**». Після успошного добавлення продажу – дані нового клієнта добавляться в таблицю. Якщо ж такий користувач існує і його загальна сума покупок більша 10000 грошових одиниць – клієнт отримує знижку 5% і загальна сума зменшується на 5%.

Далі потрібно ввести дату покупки. При натиску на зелену кнопку, йде перевірка чи все введено коректно. Якщо щось невірно, то Висвітлюється **MessageBox** в якому говориться, що невірно. Приклад можна переглянути на рис. 41.

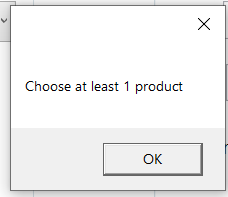


Рис. 41. MessageBox в добавленні продажу

При успішному додаванні продажу виконуються наступні дії:

* Віднімається кількість продукту в БД така, яка була введена в **TextBox** кількості. Якщо кількість в БД відповідає нулю, то воно видаляється.
* Додається клієнт в БД, якщо він не існує і доплюсовується загальна сума в його загальний рахунок покупок. Якщо клієнт існує, то лише йде добавлення загальной суми
* Висвітлюється **MessageBox** з текстом, який говорить, що продаж був успішно добавлений (Див. Рис 42).

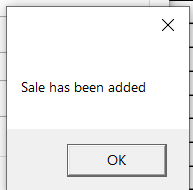


Рис. 42. MessageBox при успішному продажі

Розглянемо кнопку під номером 2. При натиску на неї перевірка чи вибраний якийсь елемент із DataGrid де в нас відображаються всі наші продажі. Якщо так воно його видаляє з таблиці «**tblNotebook**» та здійснюються наступні дії:

* Кількість яка була в продажі додається назад до **Продукту Виробника**, якщо вже цього **Продукту** не існує, то він заново доплюсовується в таблицю «**tblStorage**»
* Загальна сума продажу знімається з загальной закупки клієнта який купував в таблиці «**tblCustomers**»

Переглянемо кнопку під номером 4.1. При натиску на неї відкривається вікно (Див. Рис. 43)

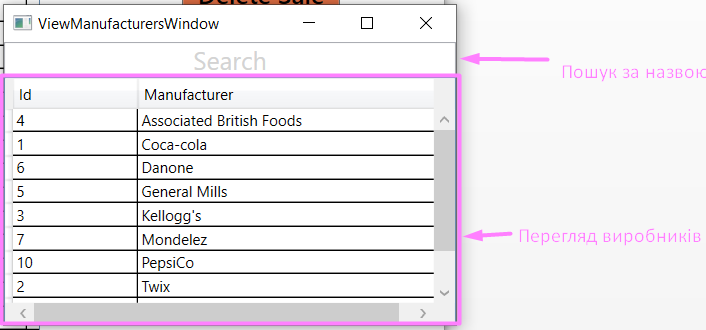


Рис. 43. Вікно перегляду Виробників

З інтрументів тут є лише **DataGrid** та **TextBox**. У **DataGrid** виводиться **унікальний ідентифікатор** та **назва Виробників** з таблиці «**tblManufacturers**». Пошук відбувається так, що незалежно від змісту введеної інформації, програма буде шукати всіх **Виробників**, по наявності конкротних знаків у їх назві (Див Рис. 44).

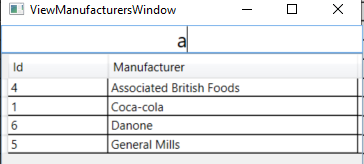
****

Рис. 44. Пошук Виробників

У **ТextBox** пошуку передбачено:

* Якщо користувач видалив звідти весь текст, то у **DataGrid** виводяться всі **Виробникі**

Переглянемо кнопку під номером 4.2. При натиску на неї відкривається вікно (Див. Рис. 45).

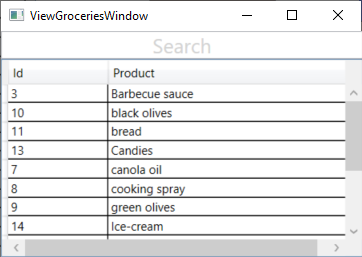


Рис. 45. Вікно перегляду Продуктів

Вікно виглядає аналогічно вікну по перегляду **Виробників**, тільки замість них тут відображаються **Продукти** з таблиці «**tblGroceries**». Пошук відбувається за таким же принципом.

Переглянемо кнопку під номером 4.3. При натиску на неї відкривається вікно (Див Рис. 46).

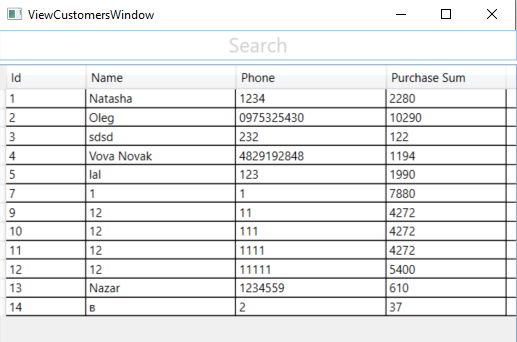
****

Рис. 46. Вікно перегляду клієнтів

Вікно має функціонал як у двох попередніх, тільки виводиться у **DataGrid** інформація про користувачів з таблиці «**tblCustomers**»

Переглянемо кнопку під номером 4.4. При натиску на неї відкривається вікно (Див. Рис. 47).

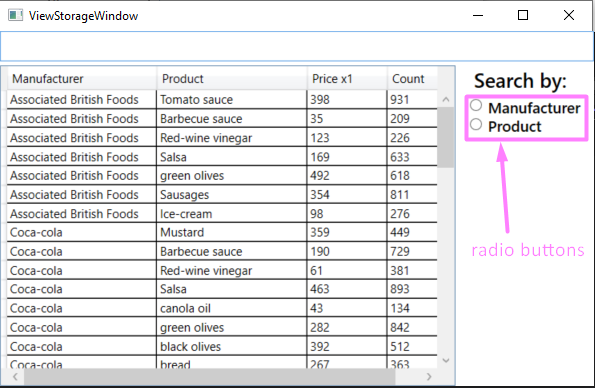


Рис. 47. Вікно перегляду Складу

Вікно переважно аналогічне трьом попереднім вікнам, але присутній один виняток. Це **RadioButtons** – елемент графічного інтерфейсу користувача, що дозволяє користувачеві вибрати тільки один з елементів набору. Ці інструменти є доповненням до пошуку. Пошук може здійснюватись за назвою: **Виробника** або **Продукту**.

Переглянемо кнопку під номером 5.1. При натиску на неї відкривається вікно (Див. Рис. 48).

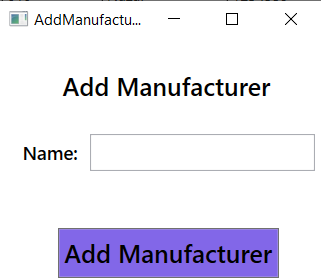


Рис. 48. Вікно добавлення нового Виробника

У текстовому полі передбачені такі позиції:

* Текстове поле пусте
* Введений **Виробник**, який вже існує в БД

Якщо текстове поле пусте і користувач натиснув на кнопку, то під текстовим полем висвітлюється червоний текст який описує, що потрібно ввести ім’я **Виробника** (Див. Рис. 49).

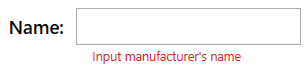


Рис. 49. Помилка при пустому текстовому полі

Якщо в текстовому полі введений користувачем **Виробник**, що вже існує в БД, то під текстовим полем висвітлюється червоний текст який описує, що такий **Виробник** вже існує в БД (Див. Рис. 50).

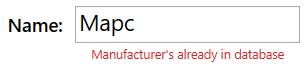


Рис. 50. Помилка при введені існуючого Виробника в БД

Якщо користувачем був введений **Виробник**, якого не існує в БД і він натиснув на кнопку – висвітлюється **MessageBox**, в якому описується, що **Виробник** був успішно доданий (Див. Рис. 51).

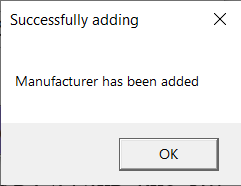


Рис. 51. MessageBox після успішного додавання Виробника

Після натиску на кнопку «ОК» на **MessageBox** – текстове поле очищується і помилка зникає. Вікно буде мати початковий вигляд як на Рис.48.

Переглянемо кнопку під номером 5.1. При натиску на неї відкривається вікно (Див. Рис. 52).

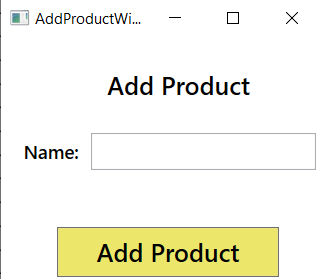


Рис. 52. Вікно добавлення нового Продукту

У текстовому полі передбачені такі позиції:

* Текстове поле пусте
* Введений **Продукт**, який вже існує в БД

Якщо текстове поле пусте і користувач натиснув на кнопку, то під текстовим полем висвітлюється червоний текст який описує, що потрібно ввести ім’я **Продукту** (Див. Рис. 53).

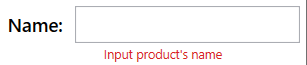


Рис. 53. Помилка при пустому текстовому полі

Якщо в текстовому полі введений користувачем **Продукт**, що вже існує в БД, то під текстовим полем висвітлюється червоний текст який описує, що такий **Продукт** вже існує в БД (Див. Рис. 54).

  
Рис. 54. Помилка при введені існуючого Продукту в БД

Якщо користувачем був введений **Продукт**, якого не існує в БД і він натиснув на кнопку – висвітлюється **MessageBox**, в якому описується, що **Продукт** був успішно доданий (Див. Рис. 55).

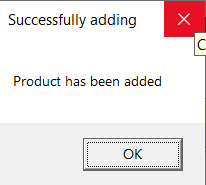


Рис. 55. MessageBox після успішного додавання Продукту

Після натиску на кнопку «ОК» на **MessageBox** – текстове поле очищується і помилка зникає. Вікно буде мати початковий вигляд як на Рис.52.

Переглянемо кнопку під номером 5.3. При натиску на неї відкривається вікно (Див. Рис. 56).

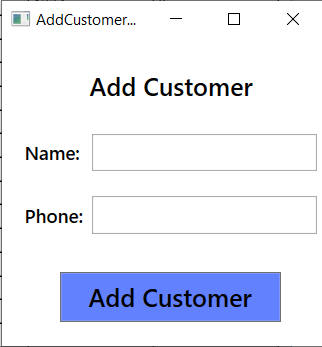


Рис. 56. Вікно добавлення нового Клієнта

Це вікно відрізняється від двох попередніх тим, що тут знаходиться два текстових поля: для ім’я клієнта, для телефону клієнта. У данному вікні передбачені такі позиції:

* Обидва текстових поля пусті;
* Текстове поле з ім’ям записане, а з номером пусте;
* Текстове поле з телефоном записане, а ім’ям пусте;
* Введений номер телефону вже зайнятий.

Якщо користувач нічого не ввів і натиснув на кнопку, то під останнім текстовим полем висвітлюються два текста які вказуються на те, що ні ім’я, ні телефон не введені (Див. Рис. 57).

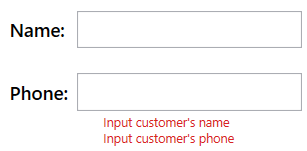
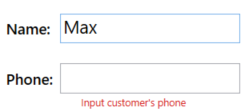


Рис. 57. Помилка при пустих текстових полях

Якщо користувач ввів лише ім’я і натинув на кнопку, то під останнім текстовим полем висвітлюється лише один текст який вказує на те, що потрібно ввести номер телефону (Див. Рис. 58).



Рисю 58. Помилка при пустому текстовому полі телефона Клієнта

Якшо користувач ввів лише телефон і натиснув на кнопку, то під останнім текстовим полем висвітлюється лише один текст який вказує на те, що потрібно ввести ім’я **Клієнта** (Див. Рис. 59).

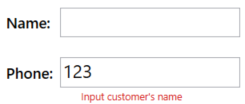


Рис. 59. Помилка при пустому текстовому полі ім’я Клієнта

Якщо користувач ввів данні в обидва текстових поля, але даний номер телефону вже зайнятий іншим **Клієнтом**, то під останнім текстовим полем висвітлюється лише один текст який вказує на те, що введний номер вже зайнятий іншим **Клієнтом** (Див. Рис. 60).

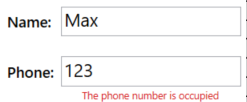


Рис. 60. Помилка при введені зайнятого телефону іншим Клієнтом

Якщо всі данні введенні і номер телефону не зайнятий і користувач натиснув на кнопку, то висвітлюється **MessageBox** в якому написано, що Клієнт успішно доданий (Див. Рис. 61).

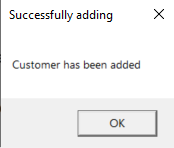


Рис. 61. MessageBox після успішного додавання Клієнта

Після натиску на кнопку «ОК» на **MessageBox** – текстові поля очищуються і помилки зникають. Вікно буде мати початковий вигляд як на Рис.56.

Переглянемо кнопку під номером 5.4. При натиску на неї відкривається вікно (Див. Рис. 62).

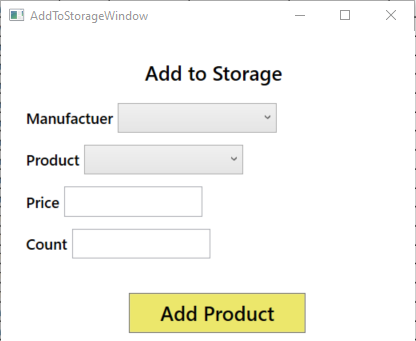


Рис. 62. Вікно добавлення нового об’єкту в Склад

Дане вікно значно відрізняється від трьох попередніх. У ньому присутні два **ComboBox** та два **TextBox**. У вікні передбаченні такі речі:

* Якщо хоча б один існтрумент пустий – буде висвітлюватись **MessageBox** вказуючий на те, що потрібно ввести;
* Якщо в **Складі** вже існує даний **Продукт** у **Виробника.**

У **ComboBox Виробника** будуть зберігатись усі **Виробники**, що є в БД. У **ComboBox Продукту** будуть зберігатись усі **Продукти**, що є в БД. У текстові поля **Ціни** та **Кількості** можна вводити лише цифри. Якщо кристувач нічого не обрав і не ввів і натиснув на кнопку то висвітлюється **MessageBox** з текстом в якому говориться, що потрібно обрати **Виробника**, **Продукт**, ввести **Ціну** та **Кількість** (Див. Рис. 63).

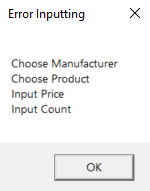


Рис. 63. MessageBox при пустих полях

Якщо користувач заповнив певні поля, а якесь пропустив. Також висвітлюється **MessageBox**, в якому буде текст в якому вказується на ті поля, які ще не заповнені. Якщо користувач Обрав **Виробника** і **Продукт** поєдання яких уже існує в БД в таблиці «**tblStorage**» - текстові поля заповнюються данними з об’єкту в БД (Див. Рис. 64).

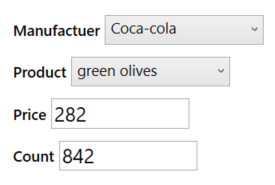


Рис. 64. Існуюче поєднання Продукту і Виробника в БД

Якщо в такому випадку користувач змінить ціну, то ціна зміниться. Якщо користувач змінить кількість, то до данної кількості в БД добавиться та кількість які користувач ввів. Всі добавлення виконаються після натиску на кнопку користувачем.При успішному заповенні всіх даних і натиску на кнопку користувачем. Висвітлюється **MessageBox** який вказує на те, що данні успішно введені в БД (Див. Рис. 64).

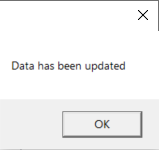


Рис. 64. MessageBox при успішному добавленні даних

Перейдемо до інструменту під номером 6. Він складається з трьох інструментів: **TextBox**, два **Image**. Текстове поле виводить максимальну кількість сторінок та на якій сторінці знаходиться користувач. На одній сторінці міститься 20 продаж. Якщо в БД знаходиться не парна кількість продаж, то добавляється ще одна сторінка в якій буде знаходитись залишок сторінок. До інструментів **Image** добавлена подія натиску на них. Якщо натиск відбувається на лівий трикутник – користувача перекидує на одну сторінку назад. Якщо користувач знаходиться на першій сторінці – його не буде перекидувати назад далі. Якщо натиск відбувається на правий трикутник – користувача перекидує на одну сторінку вперед. Якщо користувач знаходиться на останній сторінці – його не буде перекидувати далі вперед.

1. **ВИСНОВОК**

В основі кожної інформаційної системи лежить середовище доступу до інформації – система управління БД. Сьогодні важко уявити собі локальні та глобальні мережі без основної частини прикладного програмного забезпечення – бази даних та системи управління базою даних. Сфера онлайн сервісів та помічників переживає стрімкий розвиток, яка також не обходиться без БД.

В процесі виконання курсової роботи було:

* Застосовано теорію розробки інформаційної системи на практиці;
* Створена програма для ПК, яка значно полегшує запис продажів для ФОП, перегляду усіх потрібних даних для подальшої коректної роботи;
* Створено відповідну БД, встановлено властивості полів, забезпечено цілісність.

1. **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**
2. Бен Форта Освой самостоятельно SQL за 10 минут. Издание 4-е дополненное. К. : Диалектика, 2015. 288 с.
3. Бачишина Л. Д., Харів Н. О. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Бази даних» для студентів освітнього рівня бакалавр спеціальностей 113 Прикладна математика і 122 Комп’ютерні науки та інформаці технології (04- 01-13). Рівне : НУВГП, 2016. 39 с. URL: <http://ep3.nuwm.edu.ua/4062/>.
4. Туторіал по SQL. URL: <https://www.w3schools.com/sql/default.asp>.
5. Туторіал по WPF C#. URL: <https://wpf-tutorial.com/>.
6. Харів Н. О. Бази даних та інформаційні системи : навч. посіб. Рівне : НУВГП, 2018. 127. URL : <http://ep3.nuwm.edu.ua/9129/>.

**Посилання на GitHub репозиторій**

Посилання на GitHub репозиторій в якому знаходиться весь проект. URL: <https://github.com/nazaruha/Notebook_of_SoleTrader>.