ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

**КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЕКТ:**

**ПРОГРАМУВАННЯ**

на тему: "Розроблення програмного продукту для роботи з файловою базою даних про ціни на основні продукти харчування в місті"

Студента 2 курсу 6.04.121.010.19.03 групи

Спеціальності 121 "Інженерія

програмного забезпечення"

першого (бакалаврського) рівня

Мирошниченко О. С.

Керівник: професор кафедри ІС,

канд. техн. наук, доцент

Щербаков О.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка: ECTS\_\_\_\_\_\_

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Харків – 2021 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет економічної інформатики

Кафедра інформаційних систем

ЗАВДАННЯ

на комплексний курсовий проект: Програмування

студенту 2-го курсу групи 6.04.121.010.19.03

Мирошниченку Олександру Сергійовичу

1. Тема проекту: " Розроблення програмного продукту для роботи з файловою базою даних про ціни на основні продукти харчування в місті ".

2. Термін здачі студентом закінченого проекту «20» травня 2021 р.

3. Вхідні дані до проекту: літературні джерела, технічна документація щодо розроблення програм, ДСТУ з оформлення документації.

4. Зміст пояснювальної записки:

Вступ. Специфікація проекту. Програмна документація. Висновки. Додатки.

5. Перелік графічного матеріалу: UML-діаграма класів програмної системи, архітектура програмної системи



**РЕФЕРАТ**

Пояснювальна записка до курсового проекту: 41 с., 32 рис., 10 табл., 6 джерел.

Об'єктом дослідження є розробка застосунку з графічним інтерфейсом користувача

Метою роботи є розробка програмного застосунку з графічним інтерфейсом користувача для роботи з базою даних на основні продукти харчування в місті

Для досягнення поставленої мети було визначено ряд завдань, відповідно до яких було визначено необхідну архітектуру застосунку та програмні засоби

У результаті роботи було розроблено програмний застосунок з графічним інтерфейсом користувача, який працює з базою даних через фреймворк Entity Framework Core 5.0 та реалізує програму для обліку товарів.

Програмний продукт може бути використаний супермаркетами, для обліку товарів або звичайними людьми, для порівняння цін на продукти.

ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЬ, ОБЛІК ПРОДУКТІВ, WPF, SQL SERVER, ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА, ENTITY FRAMEWORK CORE

**ЗМІСТ**

[Вступ 6](#_Toc41048965)

[1 Специфікація проекту 7](#_Toc41048966)

[1.1 Постановка завдання 7](#_Toc41048967)

[1.2 Вимоги до програмного забезпечення 7](#_Toc41048968)

[2 Програмна документація 8](#_Toc41048970)

[2.1 Архітектура програмної системи 8](#_Toc41048971)

[2.2 Тестування програмної системи 11](#_Toc41048972)

[2.3 Розгортання програмного продукту 19](#_Toc41048973)

[2.4 Керівництво користувача 22](#_Toc41048974)

[2.4.1 Використання програмного продукту 22](#_Toc41048976)

[Висновки 32](#_Toc41048978)

Список використаних джерел [33](#_Toc41048979)

Додатки……………………………………………………………………………...34

**ВСТУП**

Прикладне програмне забезпечення (ППО) служить для вирішення конкретних завдань, тому воно в більшій мірі, ніж базове ПО (операційні системи, в тому числі мережні, і оболонки), індивідуалізується під певні аудиторії користувачів, орієнтоване на них і допомагає їм виконувати конкретні роботи, процеси і операції.

Будь-яке ППО функціонує в середовищі того чи іншого базового програмного забезпечення, яке, в свою чергу, керує певною інформаційною системою.

Тому, було вирішено розробити програмне забезпечення, яке може використовуватися супермаркетами, для обліку товарів або звичайними людьми, для порівняння цін на продукти.

За основу було взято найсучасніші та найтрендовіші технології, такі як .NET Core 5.0, Entity Framework Core та WPF. Це дає змогу не піклуватися про те, що в найближчий час підтримка якоїсь з технологій завершиться та доведеться переписувати значну частину коду з використанням нових бібліотек та фреймворків

1. **Специфікація проекту**
   1. **Постановка завдання**

Розроблення програмного продукту для роботи з файловою базою даних про ціни на основні продукти харчування в місті

* 1. **Вимоги до програмного забезпечення**

.NET Core 5.0, Entity Framework Core, WPF, Sql Server, Node Package Manager

1. **Програмна документація**
   1. **Архітектура програмної системи**

Було використано загальноприйняту організацію логіки програми по шарам (рис. 2.1)

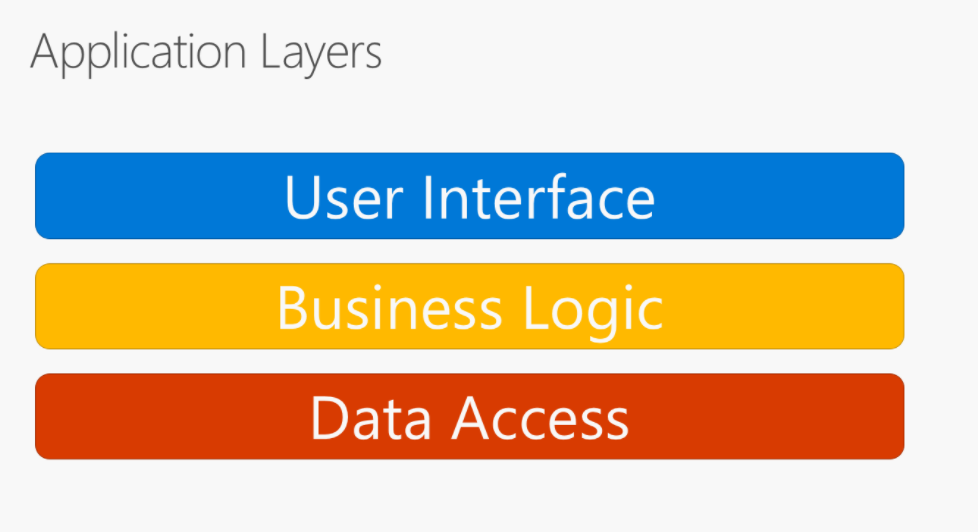


Рисунок 2.1 – Організація логіки програми

Як правило, в програмі визначаються шари призначені для користувача інтерфейсу, бізнес-логіки і доступу до даних. В рамках такої архітектури користувачі виконують запити через шар призначеного для користувача інтерфейсу, який взаємодіє тільки з шаром бізнес-логіки. Шар бізнес-логіки, в свою чергу, може викликати шар доступу до даних для обробки запитів. Шар призначеного для користувача інтерфейсу не повинен виконувати запити безпосередньо до шару доступу до даних і будь-якими іншими способами безпосередньо взаємодіяти з функціями зберігання. Аналогічним чином, шар бізнес-логіки повинен взаємодіяти з функціями зберігання тільки через шар доступу до даних. Таким чином, для кожного шару чітко визначений свій обов'язок.

Одним з недоліків традиційного багатошарового підходу є те, що обробка залежностей під час компіляції здійснюється зверху вниз. Це означає, що шар призначеного для користувача інтерфейсу залежить від шару бізнес-логіки, який, в свою чергу, залежить від шару доступу до даних. Це означає, що шар бізнес-логіки, який зазвичай містить ключові функції програми, залежить від деталей реалізації доступу до даних (і часто від наявності самої бази даних). Тестування бізнес-логіки в такій архітектурі часто утруднено і вимагає наявності тестової бази даних.

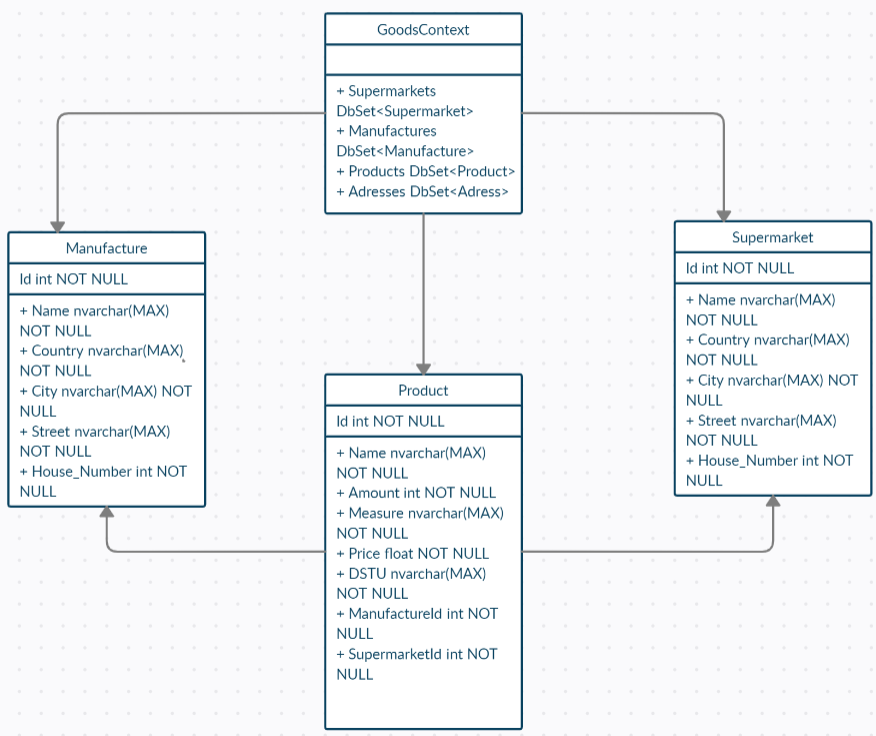


Рисунок 2.2 – Діаграма класів

На рис. 2.2 показано зв'язок класів програми.

Опис класів:

• Клас GoodsContext - контекст даних, який використовується для взаємодії з базою даних в Entity Framework Core. Цей клас містить 4 публічних поля - набори даних Supermarkets, Manufactures, Products і Addresses (Дані цих класів будуть зберігається на SQL Server).

• Клас Manufacture - описує і зберігає дані виробника продуктів харчування. У нього є 6 публічних поля - Id (первинний ключ - ідентифікатор виробника), Name (Назва виробника) та поля для зберігання інформації про адресу (Country, City, Street, HouseNumber)

• Клас Supermarket - описує і зберігає дані реалізатора (супермаркету) продуктів харчування. У нього є 3 публічних поля - Id (первинний ключ - ідентифікатор реалізатора), Name (Назва реалізатора) та поля для зберігання інформації про адресу (Country, City, Street, HouseNumber)

• Клас Product - описує один окремий продукт харчування і містить наступні поля: Id (первинний ключ - ідентифікатор продукту), Name (назва продукту харчування), Amount (кількість одиниць в супермаркеті), Measure (описує, в чому вимірюється товар), Price ( ціна за одиницю товару), DSTU (Номер державного стандарту України для конкретного товару), ManufactureId (Id виробника товару) і SupermarketId (Id реалізатора товару).

На рис. 2.3 показана архітектура програмного застосунку.

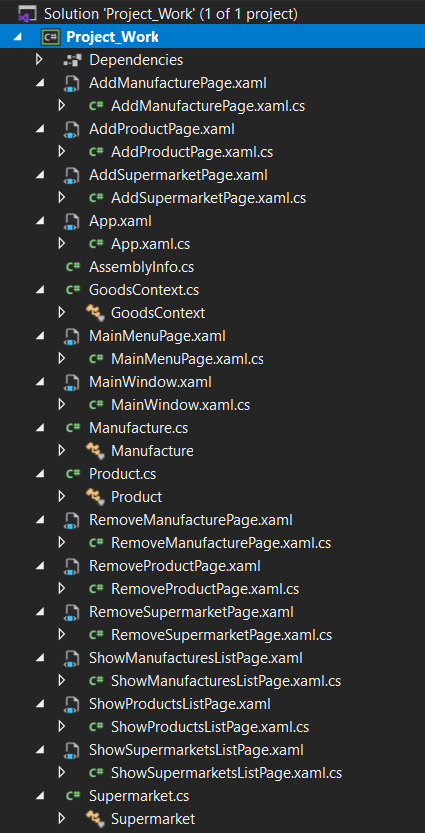


Рисунок 2.3 – Архітектура програми

При розробці даної утиліти, були використані наступні пакети (рис. 2.4):

* Microsoft.EntityFrameworkCore (для роботи с EntityFrameworkCore)
* Microsoft.EntityFrameworkCore.SqlServer (для роботи EntityFrameworkCore із сервером SQL Server)
* MaterialDesignThemes (для більш привабливішого дизайну)

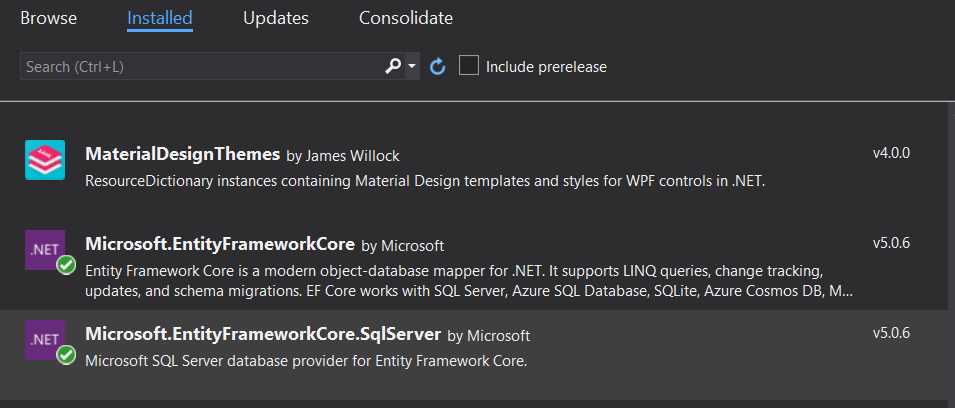


Рисунок 2.4 – Завантаженні пакети

* 1. **Тестування програмної системи**

Перелік тестів:

1. Добавлення реалізатора
2. Добавлення виробника
3. Добавлення продукту
4. Видалення реалізатора
5. Видалення виробника
6. Видалення продукту
7. Вивід даних про продукти з бази даних
8. Вивід даних про реалізаторів з бази даних
9. Вивід даних про виробників з бази даних
10. Функція завершення програми

**Тестовий приклад №1 (Добавлення реалізатора):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після додавання реалізатора, дані зберігаються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Реализатор» та вибрати «Добавить» | Відкриття нової сторінки з формою для додавання реалізатора | Відкриття нової сторінки з формою для додавання реалізатора | Так |
| 2 | Ввести усі дані та натиснути кнопку «Добавить» | З'явилося вікно з підтвердженням додавання | З'явилося вікно з підтвердженням додавання | Так |
| 3 | Перевірка таблиці Supermarkets на наявність введених даних | Наявність даних реалізатора в таблиці Supermarkets | Наявність даних реалізатора в таблиці Supermarkets | Так |

Таблиця 2.1 – Тестування добавлення реалізатора

Тест пройдено (табл. 2.1)

**Тестовий приклад №2 (Добавлення виробника):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після додавання виробника, дані зберігаються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Производитель» та вибрати «Добавить» | Відкриття нової сторінки з формою для додавання виробника | Відкриття нової сторінки з формою для додавання  виробника | Так |
| 2 | Ввести усі дані та натиснути кнопку «Добавить» | З'явилося вікно з підтвердженням додавання | З'явилося вікно з підтвердженням додавання | Так |
| 3 | Перевірка таблиці Manufactures на наявність введених даних | Наявність даних реалізатора в таблиці Manufactures | Наявність даних реалізатора в таблиці Manufactures | Так |

Таблиця 2.2 – Тестування добавлення виробника

Тест пройдено (табл. 2.2)

**Тестовий приклад №3 (Добавлення продукту):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після додавання продукту харчування, дані зберігаються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Продукт» та вибрати «Добавить» | Відкриття нової сторінки з формою для додавання продукту | Відкриття нової сторінки з формою для додавання продукту | Так |
| 2 | Ввести усі дані та натиснути кнопку «Добавить» | З'явилося вікно з підтвердженням додавання | З'явилося вікно з підтвердженням додавання | Так |
| 3 | Перевірка таблиці Products на наявність введених даних | Наявність даних реалізатора в таблиці Products | Наявність даних реалізатора в таблиці Products | Так |

Таблиця 2.3 – Тестування добавлення продукту

Тест пройдено(табл. 2.3)

**Тестовий приклад №4 (Видалення реалізатора):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після видалення реалізатора, дані про конкретного реалізатора видаляються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Реализатор» та вибрати «Удалить» | Відкриття нової сторінки з формою для видалення реалізатора | Відкриття нової сторінки з формою для видалення реалізатора | Так |
| 2 | Вибрати із списку необхідного реалізатора та натиснути кнопку «Удалить» | З'явилося вікно з підтвердженням видалення | З'явилося вікно з підтвердженням видалення | Так |
| 3 | Перевірка таблиці Supermarkets на відсутність даних | Відсутність даних реалізатора в таблиці Supermarkets | Відсутність даних реалізатора в таблиці Supermarkets | Так |

Таблиця 2.4 – Тестування видалення реалізатора

Тест пройдено(табл. 2.4)

**Тестовий приклад №5 (Видалення виробника):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після видалення виробника, дані про конкретного виробника видаляються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Производитель» та вибрати «Удалить» | Відкриття нової сторінки з формою для видалення виробника | Відкриття нової сторінки з формою для видалення виробника | Так |
| 2 | Вибрати із списку необхідного виробника та натиснути кнопку «Удалить» | З'явилося вікно з підтвердженням видалення | З'явилося вікно з підтвердженням видалення | Так |
| 3 | Перевірка таблиці Manufactures на відсутність видалених даних | Відсутність даних реалізатора в таблиці Supermarkets | Відсутність даних реалізатора в таблиці Supermarkets | Так |

Таблиця 2.5 – Тестування видалення виробника

Тест пройдено (табл. 2.5)

**Тестовий приклад №6 (Видалення продукту):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – після видалення продукту, дані про конкретний продукт видаляються у базі даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Продукт» та вибрати «Удалить» | Відкриття нової сторінки з формою для видалення продукту | Відкриття нової сторінки з формою для видалення продукту | Так |
| 2 | Вибрати із списку необхідний продукт та натиснути кнопку «Удалить» | З'явилося вікно з підтвердженням видалення | З'явилося вікно з підтвердженням видалення | Так |
| 3 | Перевірка таблиці Products на відсутність видалених даних | Відсутність даних реалізатора в таблиці Products | Відсутність даних реалізатора в таблиці Products | Так |

Таблиця 2.6 – Тестування видалення продукту

Тест пройдено(табл. 2.6)

**Тестовий приклад №7 (Вивід даних про продукти з бази даних):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – дані які знаходяться у базі даних у таблиці Products виводяться у вигляді таблиці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Списки» та вибрати «Продукты» | Відкриття нової сторінки з таблицею усіх продуктів | Відкриття нової сторінки з таблицею усіх продуктів | Так |
| 2 | Порівняння виведеної таблиці з даними у таблиці Products | Дані однакові | Дані однакові | Так |

Таблиця 2.7 – Тестування виводу даних про продукти з бази даних

Тест пройдено(табл. 2.7)

**Тестовий приклад №8 (Вивід даних про реалізаторів з бази даних):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – дані які знаходяться у базі даних у таблиці Manufactures виводяться у вигляді таблиці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Списки» та вибрати «Реализаторы» | Відкриття нової сторінки з таблицею усіх реалізаторів | Відкриття нової сторінки з таблицею усіх реалізаторів | Так |
| 2 | Порівняння виведеної таблиці з даними у таблиці Supermarkets | Дані однакові | Дані однакові | Так |

Таблиця 2.8 – Тестування виводу даних про реалізаторів з бази даних

Тест пройдено(табл. 2.8)

**Тестовий приклад №9 (Вивід даних про виробників з бази даних):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – дані які знаходяться у базі даних у таблиці Supermarkets виводяться у вигляді таблиці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Списки» та вибрати «Производители» | Відкриття нової сторінки з таблицею усіх виробників | Відкриття нової сторінки з таблицею усіх виробників | Так |
| 2 | Порівняння виведеної таблиці з даними у таблиці Manufactures | Дані однакові | Дані однакові | Так |

Таблиця 2.9 – Тестування виводу даних про виробників з бази даних

Тест пройдено(табл. 2.9)

**Тестовий приклад №10 (Функція завершення програми):**

Вимоги до тестування: програма має бути запущена

Умова проходження тесту: фактична поведінка програми збігається з очікуваною – коректним завершенням програми

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Крок сценарію | Очікуваний результат | Отриманий результат | Відмітка про проходження кроку сценарію (Так/Ні) |
| 1 | Натиснути на вкладку «Выход» | Програма одразу завершиться | Програма одразу завершиться | Так |

Таблиця 2.10 – Тестування функції завершення програми

Тест пройдено(табл. 2.10)

* 1. **Розгортання програмного продукту**

Для запуску цієї програми достатньо запустити exe-файл встановлення програми (посилання на google-диск з цим файлом у додатку А), та програма встановиться на ваш комп’ютер. На рисунках [2.5 - 2.12] показані етапи встановлення.

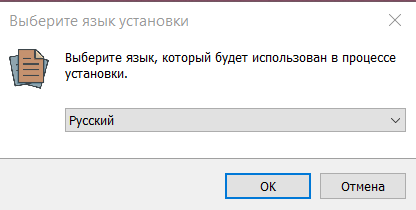


Рисунок 2.5 – Установка застосунку. Етап 1

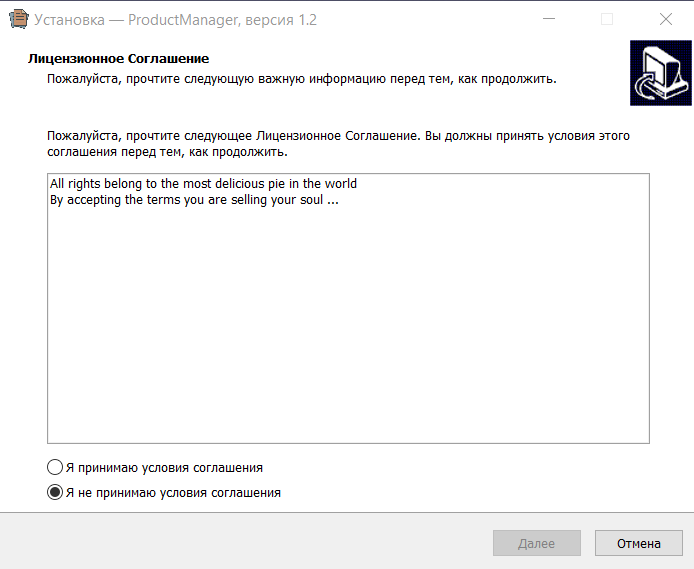


Рисунок 2.6 – Установка застосунку. Етап 2

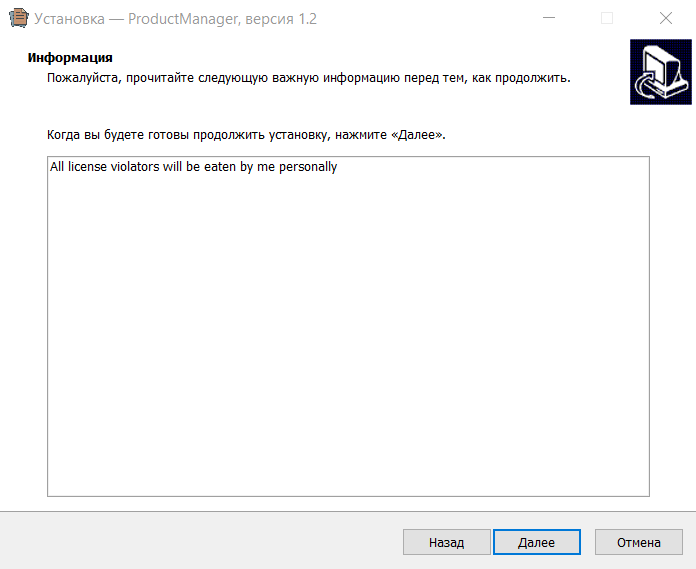


Рисунок 2.7 – Установка застосунку. Етап 3

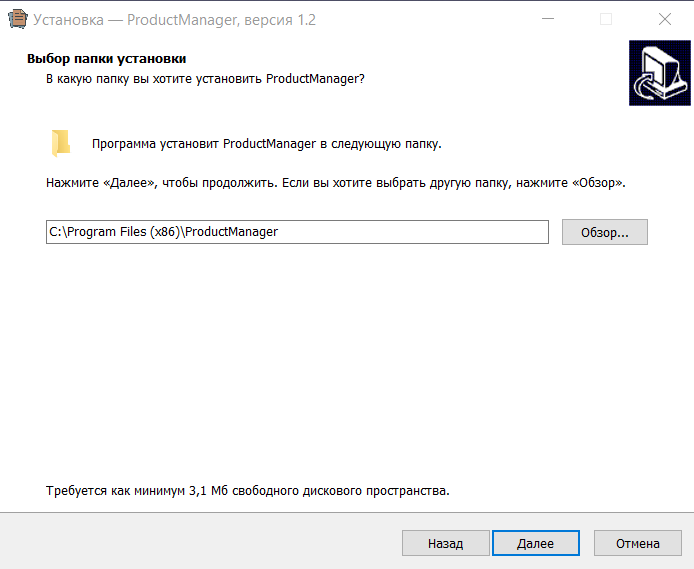


Рисунок 2.8 – Установка застосунку. Етап 4

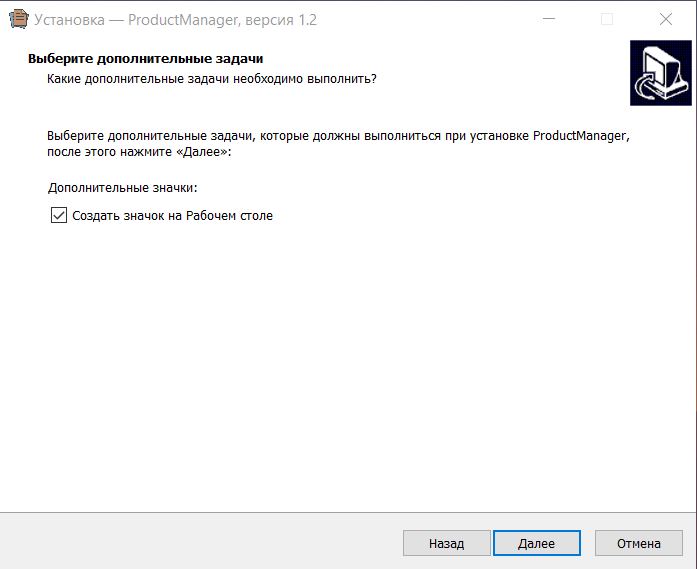


Рисунок 2.9 – Установка застосунку. Етап 5

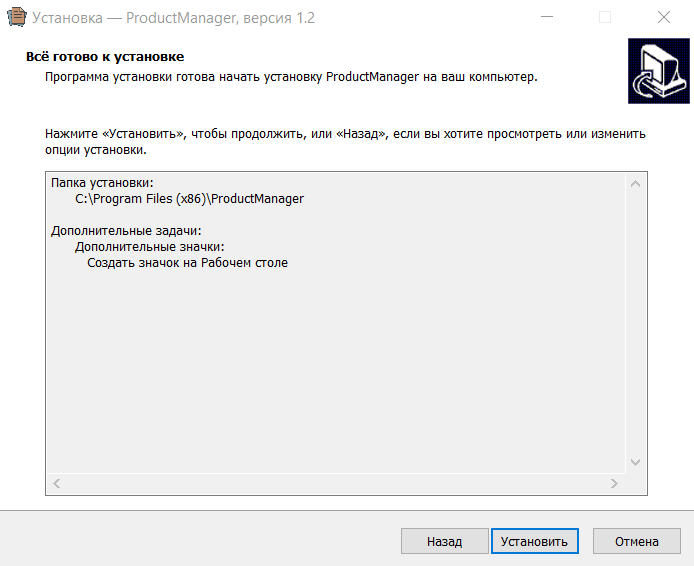


Рисунок 2.10– Установка застосунку. Етап 6

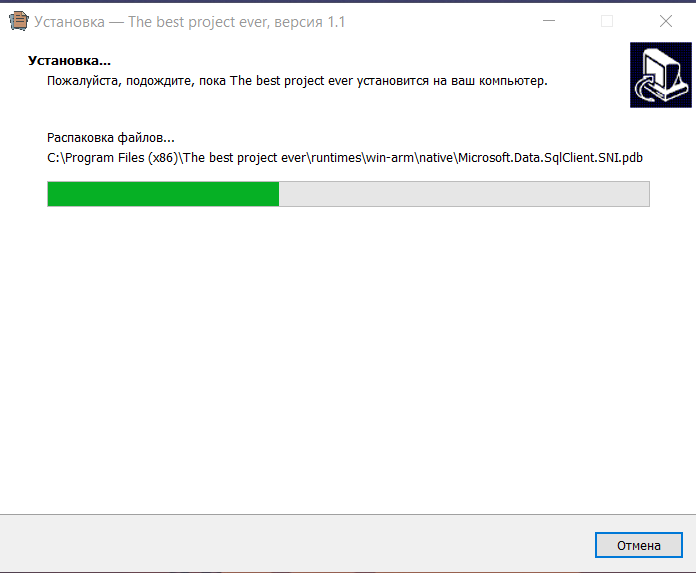


Рисунок 2.11 – Установка застосунку. Етап 7

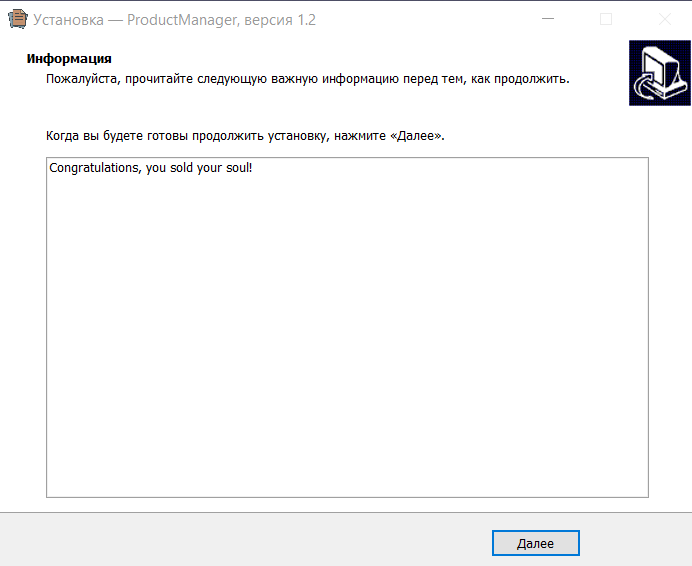


Рисунок 2.12 – Установка застосунку. Етап 8

Після цього усі необхідні файли встановляться на ваш комп’ютер та цю програму можна запустити або через ярлик на робочому столі або через exe-файл у папці у яку було встановлено дану програму

* 1. **Керівництво користувача**
     1. **Використання програмного продукту**

При запуску програмного застосунку, одразу з’являється головне меню (рис. 2.13)

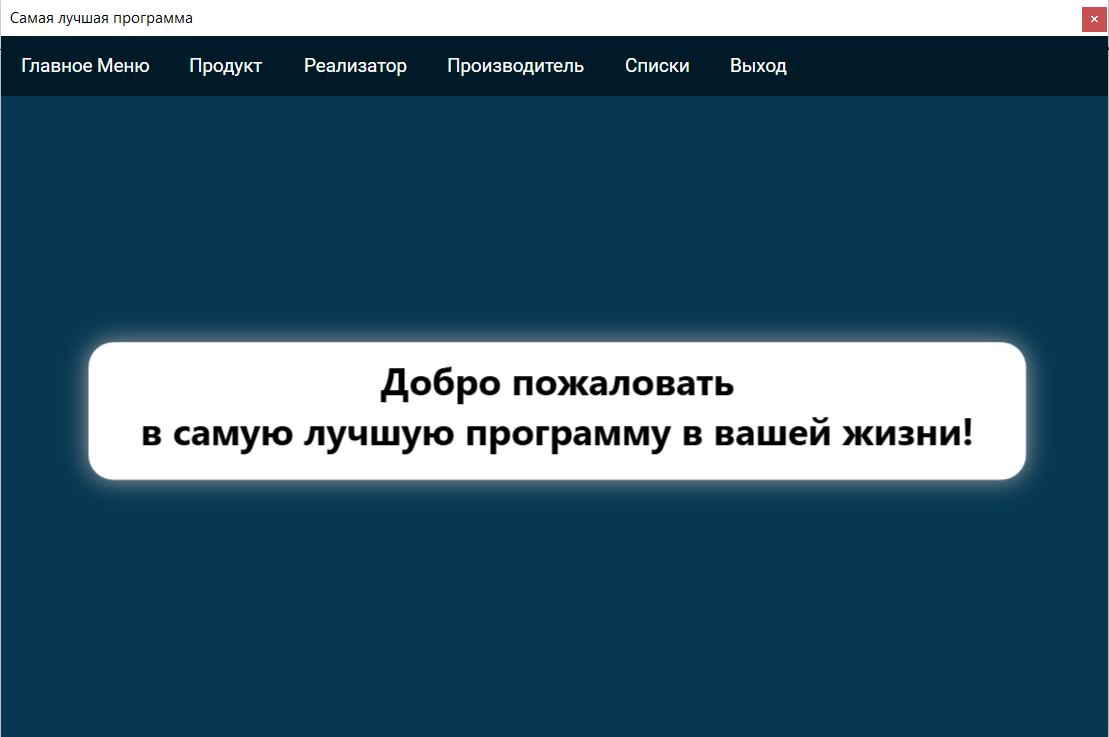


Рисунок 2.13 – Головне меню

Якщо натиснути на вкладку «Продукт», виведеться під-меню (рис. 2.14), де можна буду вибрати, що саме ми хочемо зробити з продуктами – додати новий чи видалити вже існуючий

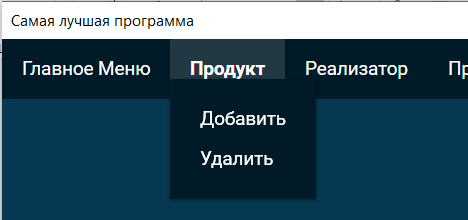


Рисунок 2.14– Під-меню пункту «Продукт»

При додаванні продукту, необхідно заповнити форму з даними(рис. 2.15), вибрати виробника та реалізатора із вже існуючих та натиснути кнопку «Добавить»

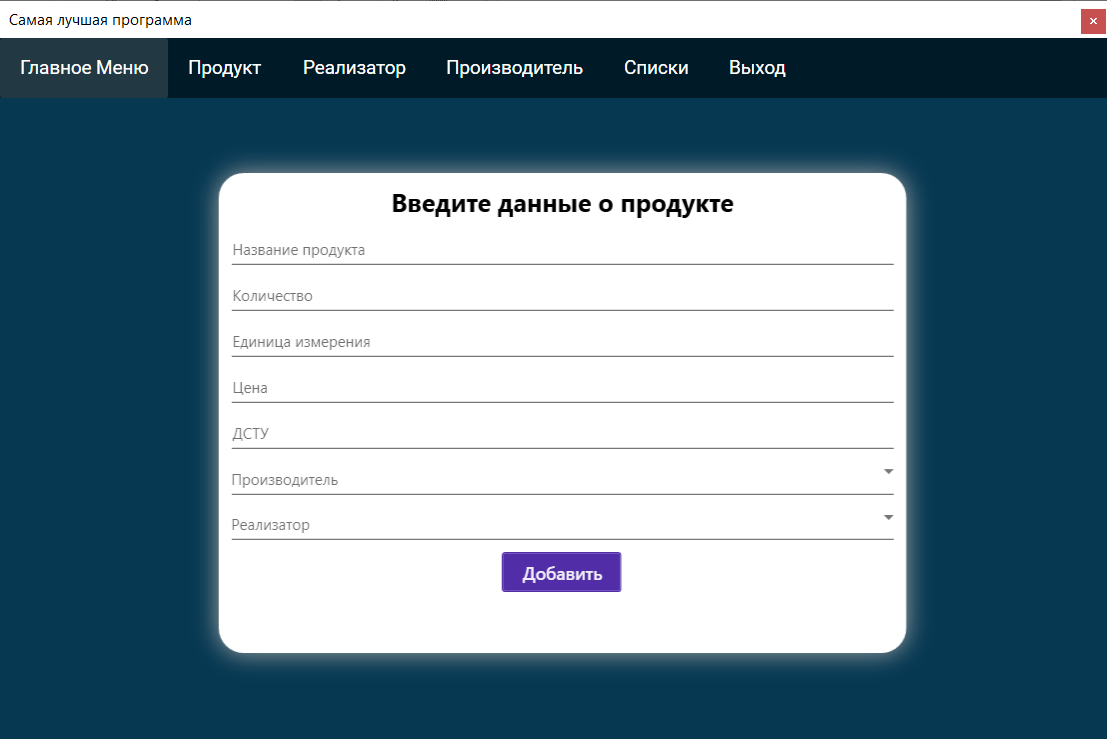


Рисунок 2.15– Вікно додавання продукту

Якщо усі дані введено правильно, то з’явиться вікно з повідомленням (рис. 2.16)

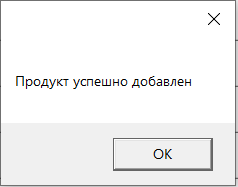


Рисунок 2.16– Повідомлення про додавання продукту

Якщо у вкладці «Продукт» вибрати «Удалить» то відкриється нова сторінка (рис. 2.17), де можна вибрати продукт, що необхідно видалити

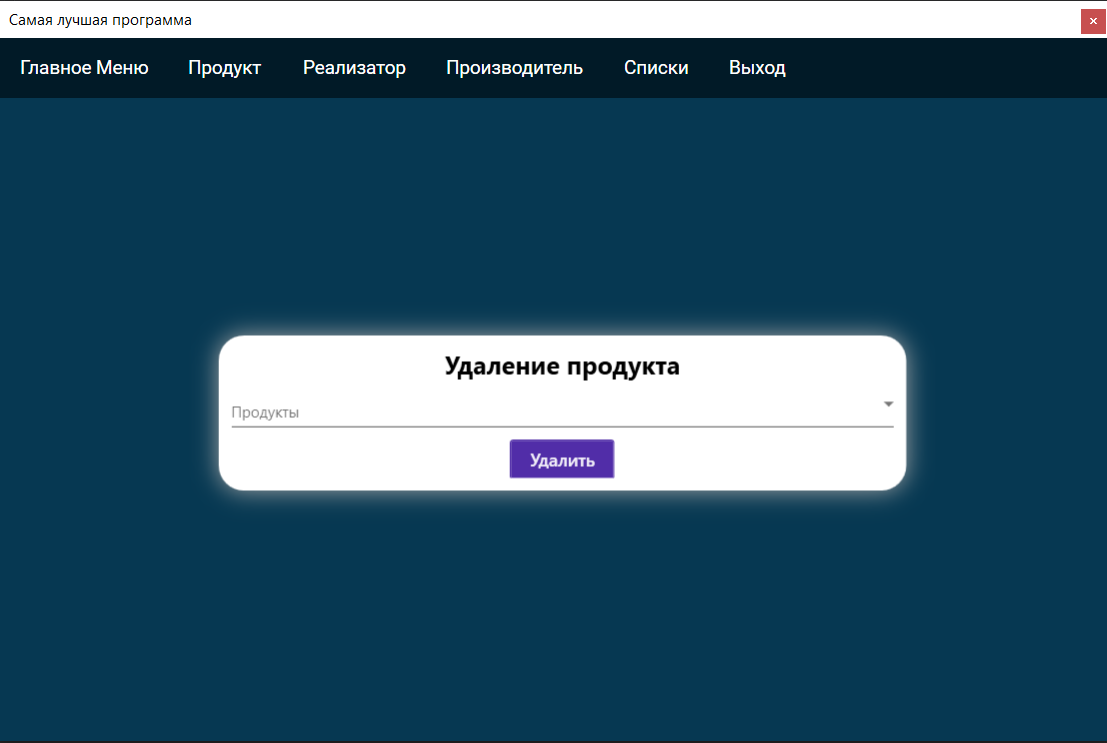


Рисунок 2.17– Вікно видалення продукту

Після вибору необхідного продукту із списку та натискання на кнопку «Удалить», з’явиться повідомлення (рис. 2.18)

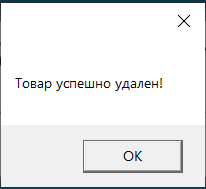


Рисунок 2.18– Повідомлення про видалення продукту

Якщо натиснути на вкладку «Реализатор», виведеться під-меню (рис. 2.19), де можна буду вибрати, що саме ми хочемо зробити з реалізаторами – додати новий чи видалити вже існуючий

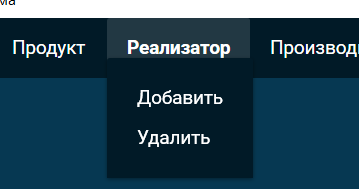


Рисунок 2.19– Під-меню пункту «Реализатор»

При додаванні реалізатора, необхідно заповнити форму з даними (рис. 2.20), та натиснути кнопку «Добавить»

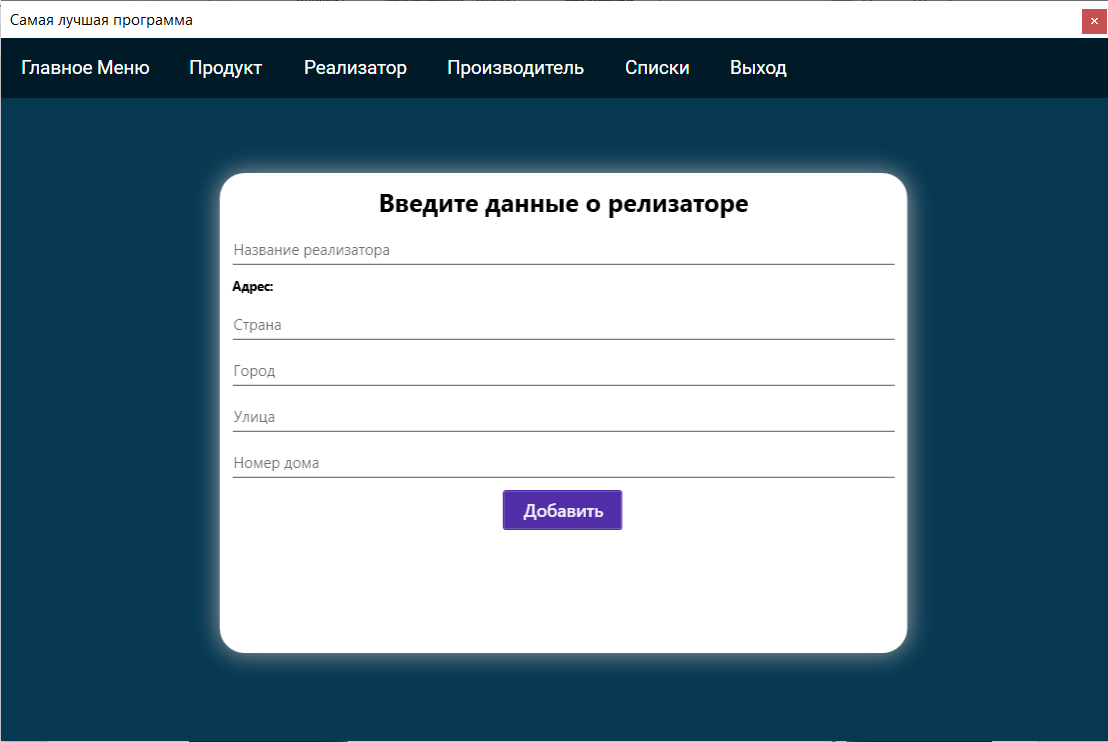


Рисунок 2.20– Сторінка додавання реалізатора

Якщо усі дані введено правильно, то з’явиться вікно з повідомленням (рис. 2.21)

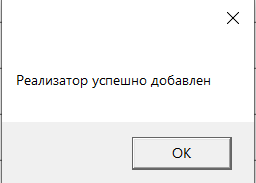


Рисунок 2.21– Повідомлення про додавання реалізатора

Якщо у вкладці «Реализатор» вибрати «Удалить» то відкриється нова сторінка (рис. 2.22), де можна вибрати реалізатора, що необхідно видалити

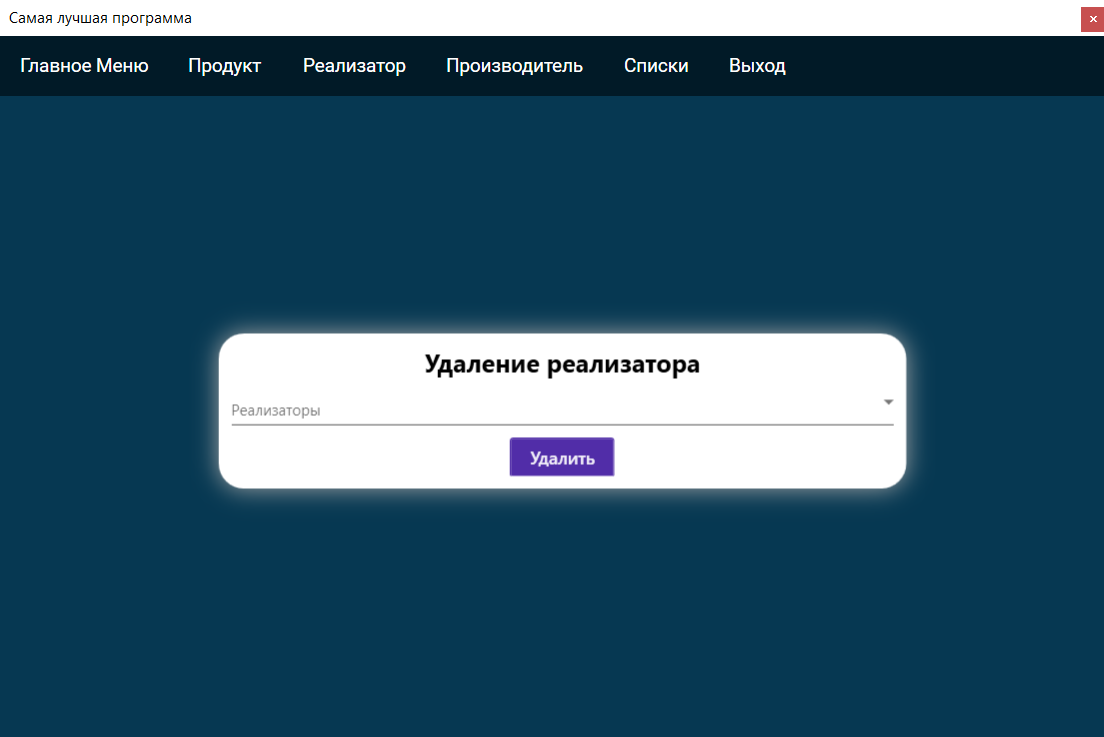


Рисунок 2.22– Сторінка видалення реалізатора

Після вибору необхідного реалізатора із списку та натискання на кнопку «Удалить», з’явиться повідомлення (рис. 2.23)

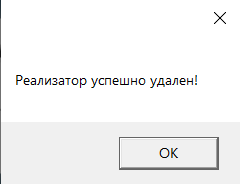


Рисунок 2.23– Повідомлення про видалення реалізатора

Якщо натиснути на вкладку «Производитель», виведеться під-меню (рис. 2.24), де можна буду вибрати, що саме ми хочемо зробити з виробниками – додати нового чи видалити вже існуючий

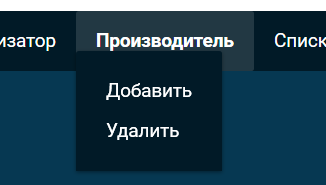


Рисунок 2.24– Під-меню пункту «Производитель»

При додаванні виробника, необхідно заповнити форму з даними (рис. 2.25), та натиснути кнопку «Добавить»

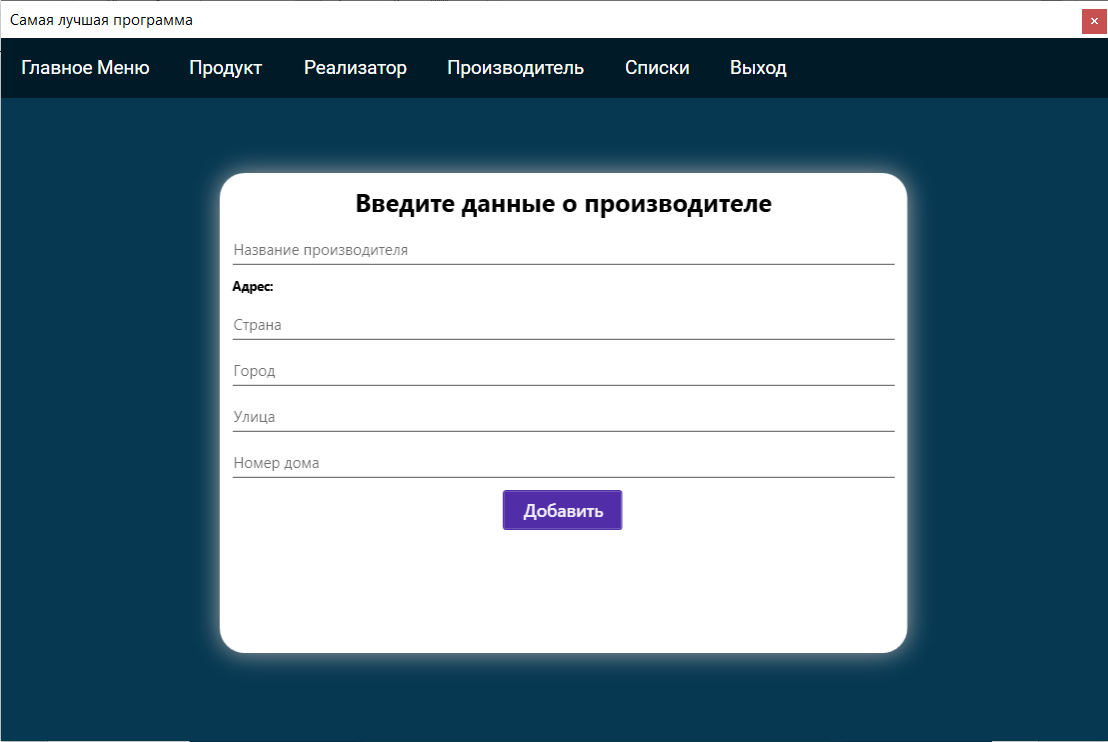


Рисунок 2.25 – Сторінка додавання виробника

Якщо усі дані введено правильно, то з’явиться вікно з повідомленням (рис. 2.26)

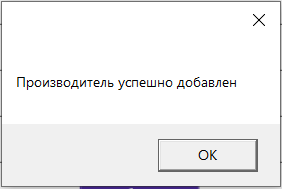


Рисунок 2.26 – Повідомлення про додавання виробника

Якщо у вкладці «Производитель» вибрати «Удалить» то відкриється нова сторінка (рис. 2.27), де можна вибрати виробника, що необхідно видалити

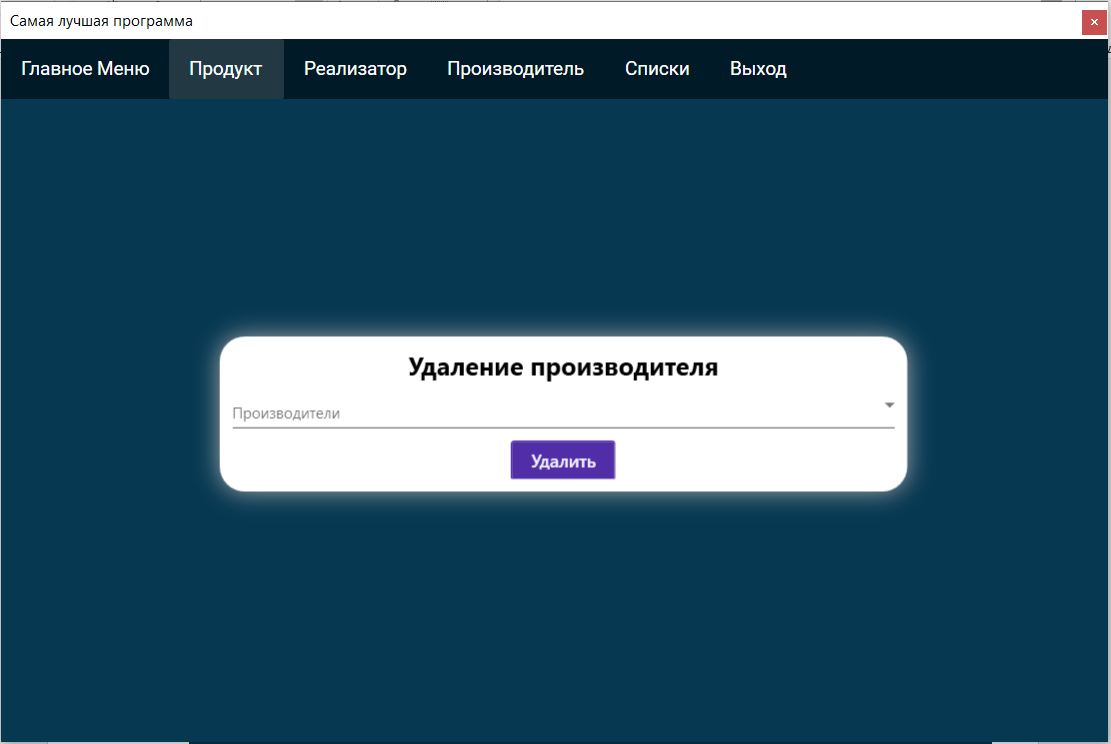


Рисунок 2.27 – Сторінка видалення виробника

Після вибору необхідного виробника із списку та натискання на кнопку «Удалить», з’явиться повідомлення (рис. 2.28)

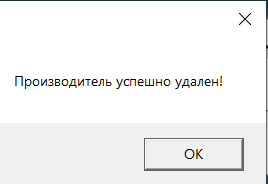


Рисунок 2.28 – Повідомлення про видалення виробника

Якщо натиснути на вкладку «Списки», виведеться під-меню (рис. 2.29), де можна буде вибрати, який саме список ми хочемо побачити – продукти, виробники чи реалізатори

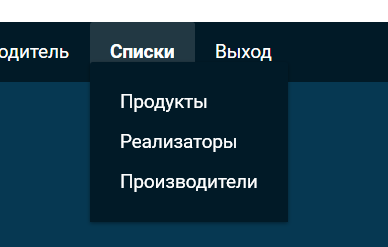


Рисунок 2.29 – Під-меню пункту «Списки»

Якщо вибрати перший пункт «Продукты», то виведеться таблиця з усіма продуктами у базі даних (рис. 2.30)

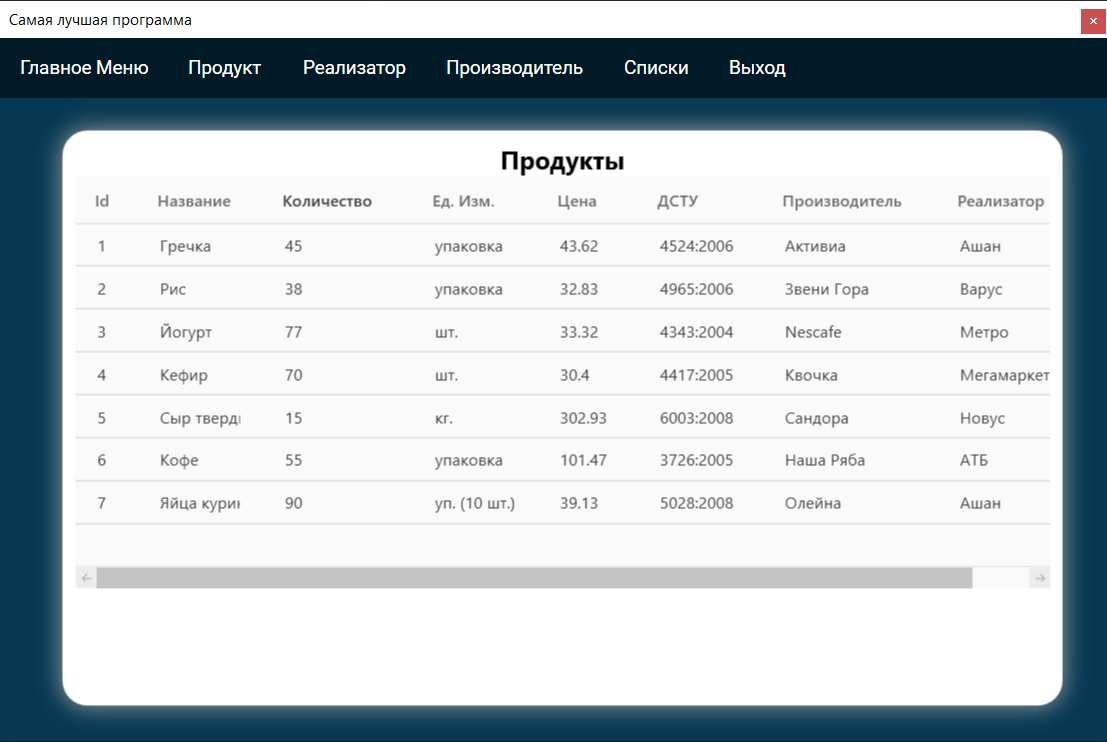


Рисунок 2.30 – Сторінка із списком продуктів

Якщо вибрати другий пункт «Производители», то виведеться таблиця з усіма виробниками у базі даних (рис. 2.31)

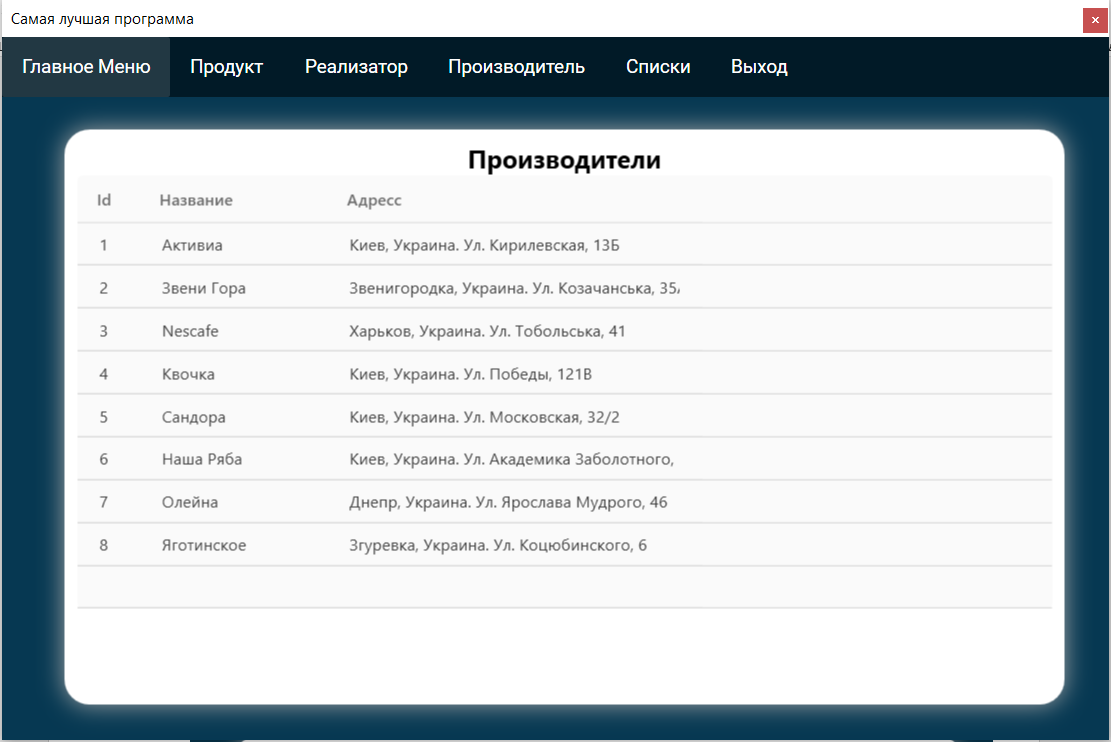


Рисунок 2.31 – Сторінка із списком виробників

Якщо вибрати третій пункт «Реализаторы», то виведеться таблиця з усіма реалізаторами у базі даних (рис. 2.32)

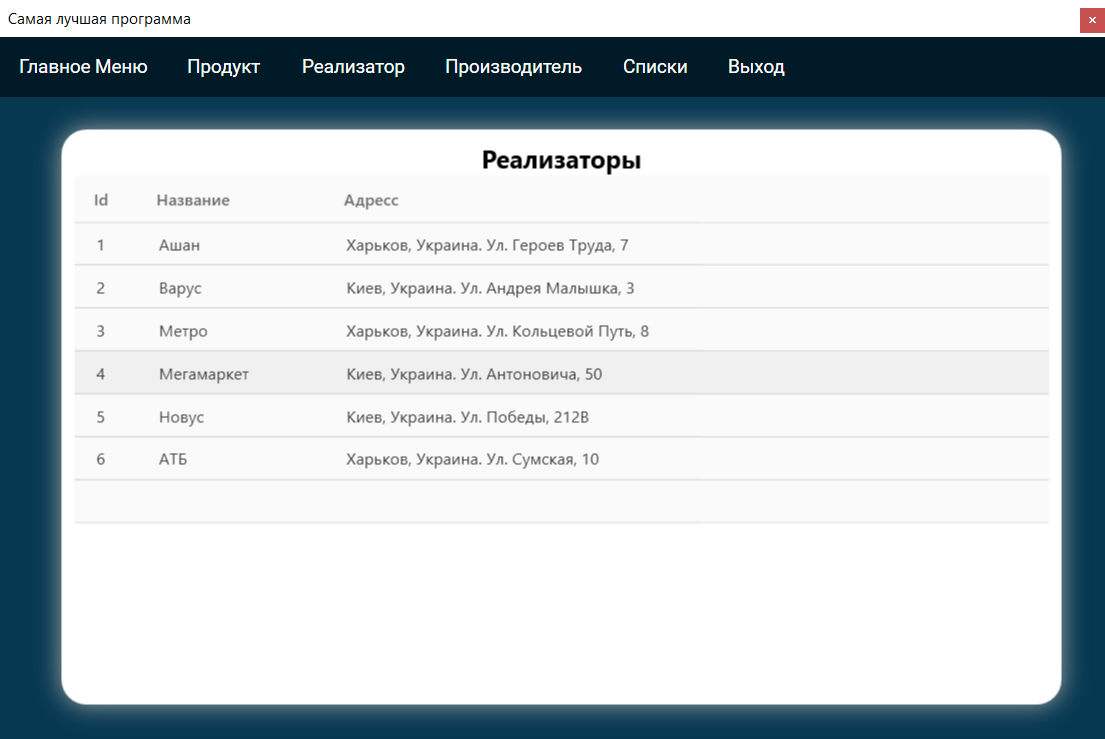


Рисунок 2.32 – Сторінка із списком продуктів

Якщо натиснути на вкладку «Выход», то програма коректно завершиться

**ВИСНОВКИ**

У ході виконання курсового завдання була створена прикладна програма з графічним інтерфейсом користувача, яка в майбутньому може буде використовуватися супермаркетами, для обліку товарів або звичайними людьми, для порівняння цін на продукти. Було отримано гарний досвід у розробці архітектури програмної системи, роботи з C#/.NET Core 5.0, Entity Framework Core, WPF та базою даних SQL Server

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Creately [Електронний ресурс] – режим доступу: [https://app.creately.com/diagram/](https://app.creately.com/diagram/create?templateId=6QI3bw0YVdo)
2. Metanit. Полное руководство по языку программирования С# 9.0 и платформе .NET 5 [Електронний ресурс] – режим доступу: <https://metanit.com/sharp/tutorial/>
3. Metanit. Руководство по Entity Framework Core [Електронний ресурс] – режим доступу: <https://metanit.com/sharp/entityframeworkcore/>
4. Metanit. Руководство по MS SQL Server 2019 [Електронний ресурс] – режим доступу: <https://metanit.com/sql/sqlserver/>
5. Metanit. Руководство по WPF [Електронний ресурс] – режим доступу: <https://metanit.com/sharp/wpf/>
6. YouTube. Изучение C# WPF на практике / Разработка программ на WPF и XAML [Електронний ресурс] – режим доступу: <https://youtube.com/playlist?list=PL0lO_mIqDDFVI0xwaYbm7h9ewYu5hftfA>

**ДОДАТКИ**

Додаток А

**Посилання на файл для встановлення програмного застосунку**

<https://drive.google.com/drive/folders/1b1tBdtzEPVHQbITDZNYDYAmVWBJy69vM?usp=sharing>

Додаток Б

**Повний код програми**

<https://github.com/AlexMyrosh/Project_Work>

Додаток В

**Код програми. Файл Manufacture.cs**

using System.ComponentModel.DataAnnotations;

namespace Project\_Work

{

class Manufacture

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string Country { get; set; }

[Required]

public string City { get; set; }

[Required]

public string Street { get; set; }

[Required]

public string House\_Number { get; set; }

public string AddressString { get; set; }

public Manufacture()

{

}

public Manufacture(string name, string country, string city, string street, string housenumber)

{

this.Name = name;

this.Country = country;

this.City = city;

this.Street = street;

this.House\_Number = housenumber;

this.AddressString = City + ", " + Country + ". Ул. " + Street + ", " + House\_Number;

}

}

}

Додаток Г

**Код програми. Файл Product.cs**

using System.ComponentModel.DataAnnotations;

namespace Project\_Work

{

class Product

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public int Amount { get; set; }

[Required]

public string Measure { get; set; }

[Required]

public double Price { get; set; }

[Required]

public string DSTU { get; set; }

[Required]

public int ManufactureId { get; set; }

public Manufacture Manufacture { get; set; }

public string ManufactureName { get; set; }

[Required]

public int SupermarketId { get; set; }

public Supermarket Supermarket { get; set; }

public string SupermarketName { get; set; }

public Product()

{

}

public Product(string name, int amount, string measure, double price, string dstu, Manufacture manufacture, Supermarket supermarket)

{

this.Name = name;

this.Amount = amount;

this.Measure = measure;

this.Price = price;

this.DSTU = dstu;

Manufacture = manufacture;

Supermarket = supermarket;

ManufactureName = manufacture.Name;

SupermarketName = supermarket.Name;

}

}

}

Додаток Д

**Код програми. Файл Supermarket.cs**

using System.ComponentModel.DataAnnotations;

namespace Project\_Work

{

class Supermarket

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string Country { get; set; }

[Required]

public string City { get; set; }

[Required]

public string Street { get; set; }

[Required]

public string House\_Number { get; set; }

public string AddressString { get; set; }

public Supermarket()

{

}

public Supermarket(string name, string country, string city, string street, string housenumber)

{

this.Name = name;

this.Country = country;

this.City = city;

this.Street = street;

this.House\_Number = housenumber;

this.AddressString = City + ", " + Country + ". Ул. " + Street + ", " + House\_Number;

}

}

}

Додаток Е

**Код програми. Файл GoodsContext.cs**

using Microsoft.EntityFrameworkCore;

namespace Project\_Work

{

class GoodsContext : DbContext

{

public DbSet<Supermarket> Supermarkets { get; set; }

public DbSet<Manufacture> Manufactures { get; set; }

public DbSet<Product> Products { get; set; }

public GoodsContext()

{

Database.EnsureCreated();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("Server=ALEXMYROSH;Database=GoodsDB;Trusted\_Connection=True");

}

}

}