Державний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Програмний модуль інформаційної системи мережі сучасних піцерій»

Студента 4 курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» Огородніка Данііла Юрійовича

_____
підпис студента

Науковий керівник
кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки Жирова Тетяна Олександрівна

_____
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки Рзаєва Світлана Леонідівна

_____
підпис гаранта

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую

Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання

на випускний кваліфікаційний проєкт студентові

Огородніку Даніілу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмний модуль інформаційної системи мережі сучасних піцерій»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту: аналіз наявних систем обліку продукції на складі ресторану, а також вивчення та аналіз методів розробки програм для управління товарами на ресторані.
Об'єкт дослідження: управління залишками продукції на ресторані.
Предмет дослідження: розробка програмного модулю управління товарами у ресторані.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Загальні положення

1.2. Змістовний опис та аналіз предметної області

1.3. Опис процесу діяльності

1.4. Аналіз успішних ІТ проектів

1.4.1. Аналіз відомих технічних рішень

1.4.2. Аналіз відомих програмних продуктів

1.5. Розробка технічного завдання до програмного забезпечення

1.5.1. Розробка функціональних вимог

1.5.2. Розробка нефункціональних вимог

1.5.3. Розробка технічного завдання

1.6. Висновок до розділу 1

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вибір середовища та інструментів розробки

2.2 Моделювання та аналіз програмного забезпечення

2.3. Архітектура програмного забезпечення

2.3.1. Опис архітектури програмного забезпечення

2.3.2. Розробка та проектування архітектури бази даних

2.3.3. Розробка та проектування архітектури проекту

2.4. Висновок до розділу 2

РОЗДІЛ 3. ОПИС ЗАПРОПОНОВАНОГО ТЕХНІЧНОГО РІШЕННЯ

3.1. Процес розробки програмного забезпечення

3.2. Огляд роботи програмного забезпечення

3.3

3.4. Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз вимог до програмного забезпечення*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Моделювання та аналіз програмного забезпечення*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Опис запропонованого технічного рішення*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	*31.05.2023*
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*	*20.06.2023*	*20.06.2023*

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту_____
Жирова Т.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми_____ Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент_____ Огороднік Д.Ю.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

(підпис, дата)

Відмітка про попередній захист

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента Огороднік Д.Ю.

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Рзаєва С.Л.

(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О. В.

(підпис, прізвище, ініціали)

«_____» _____ 20____ р.

ЗМІСТ

ВСТУП

Сучасний ринок ресторанного бізнесу характеризується жорсткою конкуренцією, що вимагає від підприємств швидкого та ефективного реагування на потреби клієнтів. Одним із чинників успіху є забезпечення якісного та швидкого обслуговування, а також ефективного управління складом та продажами товарів. З цією метою, багато ресторанів використовують різноманітні програмні засоби для автоматизації процесів управління.

Одним із найбільш актуальних напрямків розвитку програмного забезпечення є створення програмного модулю управління товарами у ресторані. Предметом дослідження є розробка програмного модулю, що дозволяє ефективно управляти складом у ресторані. Об'єктом дослідження є ресторани та їхній бізнес-процес.

Метою даного проекту є: розробка програмного модулю управління товарами у ресторані, який буде допомагати автоматизувати процеси управління складом та продажами товарів, зменшувати кількість помилок при обліку та контролю за залишками товарів, підвищувати ефективність управління рестораном в цілому та покращувати якість обслуговування.

Об’єкт дослідження: управління залишками продукції на складі ресторану.

Предмет дослідження: розробка програмного модулю управління товарами у ресторані.

Для досягнення цієї мети необхідно вирішити наступні завдання:

- проаналізувати основні процеси управління складом у ресторані;

- визначити потреби та вимоги до програмного модулю управління товарами у ресторані;
- розробити технічне завдання на розробку програмного модулю;
- вибрати оптимальні технології та інструменти для розробки програмного модулю;
- розробити програмний модуль управління товарами у ресторані;
- протестувати та впровадити програмний модуль управління товарами у ресторані;
- провести оцінку ефективності та результативності використання програмного модулю управління товарами у ресторані.

Розробка програмного модулю управління товарами у ресторані має велику практичну значущість, оскільки може допомогти ресторанам збільшити продуктивність та ефективність управління своїми бізнес-процесами. Зокрема, програмний модуль дозволить автоматизувати процеси управління складом та продажами товарів, зменшити кількість помилок при обліку та контролю за залишками товарів, забезпечить більш точну та швидку обробку замовлень клієнтів, підвищить ефективність управління рестораном в цілому.

Крім того, розробка програмного модулю управління товарами у ресторані має важливу соціальну значущість, оскільки може сприяти поліпшенню якості обслуговування. Також програмний модуль може сприяти покращенню умов роботи працівників, зменшенню їхнього навантаження та підвищенню ефективності роботи.

Тема "Розробка програмного модулю управління товарами у мережі сучасних піцерій" є досить добре вивченою в літературі та практичних дослідженнях. Багато авторів вже займалися дослідженням процесів управління складом та продажами товарів у ресторанах та розробкою

програмного забезпечення для автоматизації цих процесів. При цьому, зазвичай використовуються різноманітні підходи та технології, що можуть бути адаптовані для розробки програмного модулю управління товарами у ресторані.

Однак, не всі існуючі програмні засоби відповідають потребам та особливостям кожного конкретного ресторану, тому розробка власного програмного модулю може мати додаткові переваги у формі індивідуальних налаштувань та оптимізації процесів управління товарами для конкретної установи.

РОЗДІЛ 1
АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Розробка програмного модулю управління товарами у ресторані є актуальною задачею для бізнесу галузі громадського харчування. Такий модуль дозволяє автоматизувати процеси управління запасами, вести облік продуктів та формувати замовлення. Це дозволяє покращити ефективність бізнесу та забезпечити високу якість обслуговування клієнтів.

На ринку існує декілька програмних додатків для управління запасами та продуктами у ресторанах. Однак, багато з них мають недоліки, такі як обмежені можливості, складний інтерфейс, низьку швидкість роботи та високу вартість. Крім того, деякі додатки не підтримують міжнародні стандарти та мови, що обмежує їхнє застосування.

Перші програмні додатки для управління запасами та продуктами у ресторанах з'явилися більше десяти років тому. Вони мали обмежені можливості та були доступні тільки на локальних серверах. Однак, з появою хмарних технологій та розвитком мобільних пристроїв, на ринку з'явилися нові додатки з більш розширеними можливостями та доступнішою ціною. Сьогодні на ринку можна знайти декілька десятків додатків для управління товарами у ресторанах, які надають різноманітні функції, включаючи облік запасів, меню, замовлення та відгуки клієнтів.

1.2 Змістовний опис та аналіз предметної області

POS системи - це комплекс спеціального обладнання, призначеного для прискорення роботи касирів/операторів за рахунок її автоматизації. Основою будь-якої POS системи для магазину чи ресторану є

спеціалізований комп'ютер з прикладним програмним забезпеченням або додаток, де обробляються всі необхідні дані. В цілому, POS системи поділяються на хмарні та стаціонарні [1].

Стаціонарна POS-система - це програмне забезпечення, яке встановлюється на конкретний пристрій, наприклад, на комп'ютер або на термінал платежів у ресторані. Ці системи зазвичай підключаються до інших пристроїв, таких як принтери чеків та сканери штрих-кодів. Системи, які встановлюються на пристрої, можуть бути належним чином налаштовані та настроєні під потреби ресторану [2].

Переваги стаціонарних POS-систем [3]:

- Вони зазвичай мають більше функцій, ніж хмарні POS-системи.
- Вони зазвичай працюють швидше та більш надійно, оскільки вони не залежать від Інтернет-з'єднання.
- Ресторан може зберігати дані локально та не залежати від сторонніх сервісів, таких як хмарні сервери.

Недоліки стаціонарних POS-систем [3]:

- Ресторан повинен придбати та налаштувати обладнання для стаціонарної POS-системи, що може бути витратним.
- Вони не є настільки мобільними, як хмарні POS-системи, оскільки їх можна використовувати лише з конкретного пристрою.

Хмарна POS-система - це програмне забезпечення, яке працює в хмарі та може бути доступним через Інтернет. Це означає, що ресторан може використовувати POS-систему з будь-якого місця, де є Інтернет-з'єднання. Хмарні POS-системи зазвичай мають багато функцій, які дозволяють ресторану контролювати замовлення, запаси та фінанси. Хмарні POS-системи можуть бути прості, які просто надають функції обліку продажів та керування запасами, або комплексні, які включають в себе функції з

управління замовленнями, резервуванням столиків, винним списком та інші функції. Деякі з найбільш популярних хмарних POS-систем на ринку включають TouchBistro, Square, Toast, Lightspeed та ShopKeep [2].

Переваги хмарних POS-систем [3]:

- Ресторан може використовувати хмарну POS-систему з будь-якого місця, де є Інтернет-з'єднання.
- Хмарні POS-системи зазвичай забезпечують безпеку даних та забезпечують резервне копіювання даних.
- Ресторан не потребує додаткового обладнання для використання хмарної POS-системи.

Недоліки хмарних POS-систем [3]:

- Вони можуть залежати від якості та стабільності Інтернет-з'єднання.
- Ресторан повинен платити за користування хмарною POS-системою на постійній основі, що може бути витратним на довгу перспективу.
- Деякі хмарні POS-системи можуть бути менш надійними, ніж стаціонарні POS-системи, оскільки вони залежать від сторонніх сервісів та серверів.

В цілому, обидві системи мають свої переваги та недоліки, тому вибір залежить від потреб ресторану та його можливостей. Сучасні ресторани намагаються бути більш мобільними та гнучкими, саме тому, хмарна POS-система може бути кращим вибором, навіть не зважаючи на більші витрати у довгостроковій перспективі, порівняно із стаціонарними POS-системами.

1.3 Опис процесу діяльності

Першим етапом роботи буде встановлення додатку на персональний комп’ютер або на мобільний пристрій (телефон, планшет).

Після реєстрації клієнт може налаштувати програму згідно своїх потреб, додати інгредієнти, що зберігаються на складі ресторану, налаштувати меню, списання інгредієнтів. Для кожного інгредієнту є можливість додати перелік певних параметрів, таких як собівартість, поточні залишки на складі, назва та одиниці виміру товарів. Налаштування меню дозволить списувати або продавати окремі конкретні страви, внаслідок чого будуть списуватись інгредієнти у заданих грамовках зі складу ресторану.

Після того як клієнт налаштував програму та підготовив до роботи, він може прийняти замовлення у свого клієнта та списати конкретну страву на замовлення. Система збереже ці дані, обробить інформацію по списанням інгредієнтів зі складу та в майбутньому зможе надати коректну статистику щодо реальних залишків продукції на складі ресторану.

Оскільки система запам’ятовує усі списання продукції, записує їх причини та кількість списаного, а також автоматично оновлює дані про залишки товарів та фінансову складову, користувач у будь-який час зможе перевірити поточні залишки, подивитись тенденцію замовлень та, як приклад, провести переоблік, звіряючи дані, надані програмою з реальними залишками на складі.

Також, однією із головних функцій POS-системи є звіти. Вони автоматично формуються на основі дій у програмі, таких як списання, продаж та поставки товарів. Звіти відображаються у призначених для цього пунктах меню, полегшуючи керування та контроль над залишками та продажами.

Схема алгоритму оновлення даних представлена на рисунку 1.1.

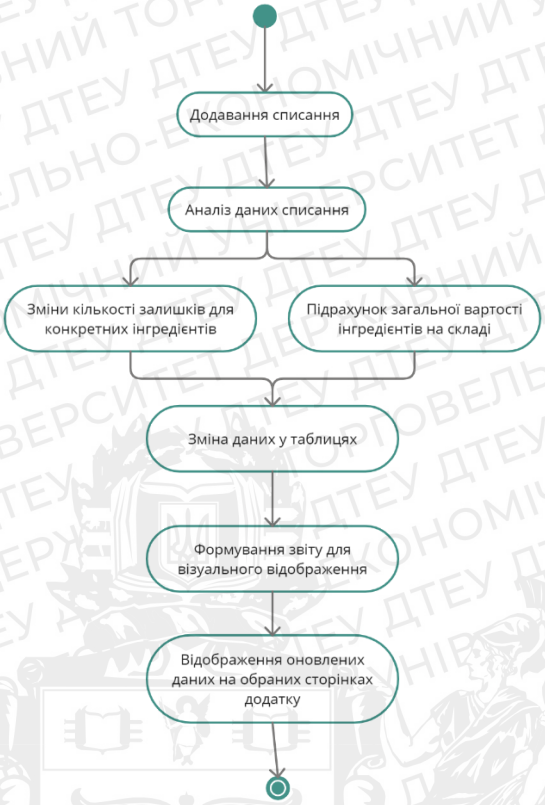

Рис 1.1 Схема структури діяльності процесу оновлення даних таблиць

1.4 Аналіз успішних IT проектів

1.4.1 Аналіз відомих технічних рішень

POS-системи в даний час дуже популярні в бізнес-середовищі, і розробники постійно працюють над поліпшенням своїх продуктів. Нижче наведено приклади відомих технічних рішень:

— Операційна система Android. Android є операційною системою, яка забезпечує зручне та ефективне використання хмарних POS-систем на мобільних пристроях. Вона є безкоштовною та має велику кількість розробників, які створюють програми для неї. Однак, підтримка української мови в Android не є повноцінною, що може створити труднощі для користувачів з України [5].

— Протокол Bluetooth. Bluetooth є бездротовим протоколом, який дозволяє з'єднувати хмарні POS-системи з мобільними пристроями через бездротову мережу. Він є безкоштовним та доступним у всіх країнах, включаючи Україну. Однак, протокол Bluetooth може мати обмеження в зоні покриття та швидкості передачі даних [6].

— Протокол Wi-Fi. Wi-Fi є бездротовим протоколом, який забезпечує високу швидкість передачі даних та великий діапазон покриття. Він є безкоштовним та доступним у всіх країнах, включаючи Україну. Однак, протокол Wi-Fi може мати проблеми з безпекою та вимагати наявності точки доступу до Інтернету [7].

— API. API (Application Programming Interface) дозволяє розробникам зв'язувати хмарні POS-системи з іншими додатками та сервісами. Він є безкоштовним та доступним у всіх країнах, включаючи Україну. API дозволяє збільшувати функціональність та можливості хмарних POS-систем, однак вимагає відповідної інтеграції та налагодження [8].

В таблиці 1.2 показане порівняння технологій за доступністю в Україні та вартістю використання.

Таблиця 1.2.

Порівняння технологій за доступністю в Україні та вартістю використання.

Назва технології	**Підтримка української мови**	**Можливість безкоштовного користування**	**Доступність в Україні**
Android	Не повна	Є	Є
Bluetooth	Є	Є	Є
Wi-Fi	Є	Є	Є
API	Є	Є	Є

1.4.2. Аналіз відомих програмних продуктів

Існує чимало готових програмних продуктів, які надають можливість керувати залишками на складі ресторану.

— Square. Square є однією з найпопулярніших хмарних POS-систем у світі, яка пропонує безкоштовний POS-термінал та низькі комісії за операції. Вона підтримує українську мову та доступна в Україні. Однак, для деяких функцій Square може вимагати додаткову плату, і її не варто використовувати для великих підприємств [11].

— Shopify. Shopify є хмарною POS-системою, яка дозволяє створювати онлайн-магазини та працювати зі звичайними терміналами. Вона підтримує українську мову та доступна в Україні, але може бути дорожчою в порівнянні з іншими хмарними POS-системами [12].

— Toast. Toast є хмарною POS-системою, яка підходить для ресторанів та інших закладів харчування. Вона підтримує українську мову та доступна в Україні, але вимагає плату за використання деяких функцій та може бути дорожчою в порівнянні з іншими хмарними POS-системами [8].

— Lightspeed. Lightspeed є хмарною POS-системою, яка підходить для різних видів бізнесу. Вона підтримує українську мову та доступна в Україні, але може бути дорожчою в порівнянні з іншими хмарними POS-системами [9].

— Loyverse. Loyverse є безкоштовною хмарною POS-системою, яка підходить для різних видів бізнесу, зокрема ресторанів, кафе та магазинів. Вона підтримує українську мову та доступна в Україні, проте деякі функції можуть вимагати плату [10].

У таблиці 1.3 наведено порівняння популярних програмних продуктів.

Таблиця 1.3.

Порівняння популярних програмних продуктів.

Програмний продукт	**Переваги**	**Недоліки**	**Підтримка української мови**	**Можливість безкоштовного користування**
Square	Безкоштовний POS-термінал; низькі комісії; підтримка української	Додаткова плата за деякі функції; не підходить для великих підприємств	Є	Є
Shopify	Можливість створення онлайн-магазинів та робота з терміналами; підтримка української мови	Дорожча порівняно з іншими POS-системами	Є	Ні
Toast	Підходить для ресторанів та інших закладів харчування; підтримка української мови	Вимагає плати за використання деяких функцій; може бути дорожча	Є	Ні
Lightspeed	Підходить для різних бізнесів; підтримка української мови	Може бути дорожча	Є	Ні
Loyverse	Безкоштовна; підходить для різних бізнесів; підтримка української мови	Деякі функції можуть вимагати плату	Є	Є

Проаналізувавши ряд вище вказаних програмних продуктів можна виявити переваги на недоліки існуючих рішень. Хоча кожна з хмарних POS-систем має свої переваги та недоліки, в основному, усі ці програми є платними, іноді є змога спробувати роботу на безкоштовній версії, однак у такому випадку сильно применшується функціонал. Тому ідея створення безкоштовного програмного продукту для обліку товарів на складі є актуальною.

1.5 Розробка технічного завдання до програмного забезпечення

1.5.1 Розробка функціональних вимог

Діаграма варіантів використання зображена на рисунку 1.4.

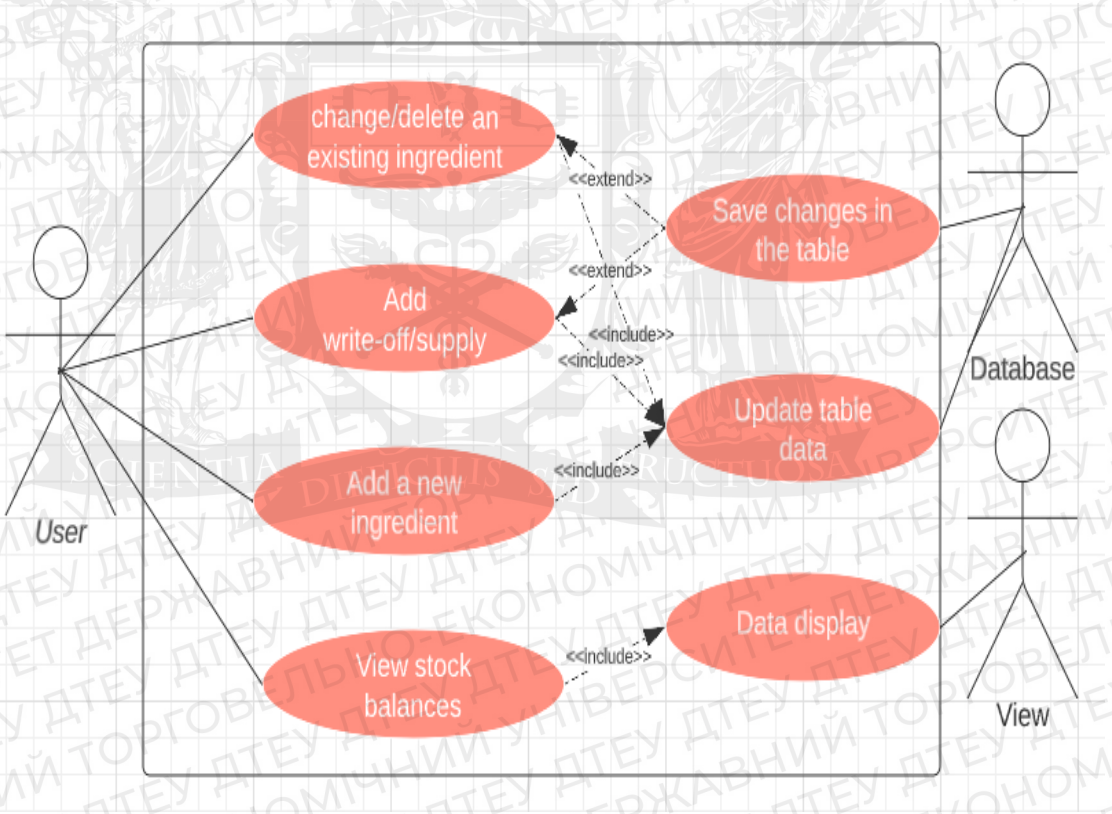

Рис 1.2 – Діаграма варіантів використання

Таблиця 1.4.

Функціональні вимоги та їх пріоритети

Варіант використання	Функціональна вимога	Пріоритет
Перегляд залишків	Відображення залишків продукції на складі ресторану	Високий
Списання товару	Списання товару зі складу і оновлення кількості залишків товарів на складі	Високий
Додавання поставок	Додавання нових поставок товарів на склад і оновлення кількості залишків товарів на складі;	Високий
Продаж страв	Продаж страв та оновлення кількості залишків товарів на складі, які входять до складу страви	Середній

1.5.2 Розробка нефункціональних вимог

— Були виділені наступні нефункціональні вимоги:

— Продукт повинен працювати на операційній системі Windows;

— Користувачі повинні мати доступ до програми з різних місць;

— Програмне забезпечення має мати інтуїтивний і зрозумілий інтерфейс для користувачів;

— Додавання і списування товарів має бути відбито в системі миттєво, щоб користувач міг бачити оновлені дані;

— Програмне забезпечення має працювати швидко і ефективно навіть при великих обсягах даних.

1.5.3 Розробка технічного завдання

Призначенням розробки є програмне забезпечення, яке дозволяє керувати залишками на складі та переглядати поточну інформацію про них.

Метою даного проекту є розробка програмного модулю управління товарами у ресторані, який буде допомагати автоматизувати процеси управління складом та продажами товарів, зменшувати кількість помилок при обліку та контролю за залишками товарів, підвищувати ефективність управління рестораном в цілому та покращувати якість обслуговування.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

— Обрати мову програмування, якою буде написаний додаток;

— Обрати базу даних для збереження даних;

— Програмне забезпечення має бути розроблене з використанням принципів SOLID і шаблонів проектування;

— Програмне забезпечення повинно бути оптимізоване для швидкої роботи з великими обсягами даних;

— Обрати технологію розробки інтефейсу;

— Створити функцію відображення залишків продукції на складі ресторану. Функція повинна коректно відображати дані для перегляду.

— Створити функції додавання, списання та редагування списань продукції зі складу. Функції повинні швидко змінювати дані у базі даних.

— Створити функцію створення страв. Функція повинна враховувати усі інгредієнти, які входять у страву та вносити зміни у відповідні поля інгредієнтів, в наслідок роботи зі стравами.

— Розробити зручний та інтуїтивно зрозумілий інтерфейс користувача;

— Створити базу даних для зберігання інформації. Розробити фізичну та логічну модель бази даних.

1.6 Методи тестування додатку «StoreHouse»

Випробування програмного забезпечення звичайно проводиться на різних етапах його життєвого циклу. Процес випробування включає такі кроки:

- Визначення функціональності, яка підлягає випробуванню, а також того, що не підлягає випробуванню.
- Формулювання підходів, які будуть використовуватись для даного продукту.
- Написання тест-кейсів.
- Розробка критеріїв для визначення успішності випробувань.
- Визначення вимог до середовища, в якому буде проводитись випробування.
- Проведення випробування та оцінка результатів.
- Підготовка звіту з результатами.

Системні вимоги для програми:

- Операційна система повинна бути Windows 10 або більш нова версія.
- Потрібна наявність SQL Server 2019.

Апаратні вимоги:

- Комп'ютер повинен мати достатню кількість вільної пам'яті для встановлення програми.
- Процесор повинен мати тактову частоту 1 ГГц і підтримувати 32 або 64-розрядну архітектуру.
- Оперативна пам'ять повинна бути не менше 1 ГБ.

Випробування проводилось на комп'ютері з процесором Intel® Core i5-9600K, тактовою частотою 3.7GHz, операційною системою Windows 10 і за наявності програми SQL Server 2019.

1.7 Висновки до розділу

В результаті проведення аналізу вимог до розробки програмного модулю інформаційної системи мережі сучасних піцерій, було проведено дослідження предметної області – POS-системи. На даний момент вони

досить розповсюджені у сфері ресторанного бізнесу. Наразі системи вдосконалюються, аби надати більш якісне та дешеве програмне забезпечення для своїх клієнтів.

Було досліджено процес діяльності подібних програм. В основному він поділяється на такі етапи: встановлення додатку на персональний комп'ютер, налаштування програми, користування програмою(перегляд залишків на складі ресторану, додавання списань, постачань, інгредієнтів).

Проведено дослідження готових технологічних рішень та успішних IT – проектів (Square, Shopify, Toast, Lightspeed, Loyverse). Виявлено, що більшість з них можуть надати якісні послуги по відслідковуванню залишків на складі та їх обліку. Однак майже всі подібні програми передбачають сплату за можливість доступу до всіх функцій. Тому було вирішено створити безкоштовний продукт із функціями управління товарами на складі ресторану.

Сформульовані головні задачі розробки програмного забезпечення: перегляд залишків на складі ресторану, управління списаннями, постачаннями, робота з меню ресторану).

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір середовища та інструментів розробки

Зараз існує безліч різних середовищ для розробки програмного забезпечення, що підтримують мову програмування C#. Для нашої розробки ми вибрали Microsoft Visual Studio з пакетом розширень ReSharper. Це інтегроване середовище розробки програмного забезпечення, розроблене компанією Microsoft, яке дозволяє створювати різноманітні програми, такі як програми з графічним інтерфейсом (Windows Forms, Windows Presentation Foundation), консольні програми та веб-додатки [13]. Крім того, Visual Studio підтримує такі мови програмування, як Visual C#, Visual Basic, Visual F#, Visual C++, Python та інші [14]. Це середовище розробки також підтримує розробку додатків не тільки для платформи Windows, але і для популярних мобільних платформ, таких як Android та iOS. Обране середовище розробки підтримує рефакторинг коду, включаючи операції, такі як інтелектуальне перейменування змінних, вилучення коду в новий метод та зміну порядку параметрів методів. Іншою корисною функцією, вбудованою в Visual Studio, є "IntelliSense" [15] (див. рис. 2.1).

Для збереження даних була обрана база даних SQL Server. SQL Server (Structured Query Language Server) - це реляційна база даних, яку розробляє та підтримує корпорація Microsoft. Вона забезпечує зберігання та обробку даних, а також надає інструменти для їх аналізу та управління. SQL Server підтримує різноманітні операції з даними, такі як додавання, видалення, оновлення, вибірка, об'єднання даних з різних таблиць тощо [16].

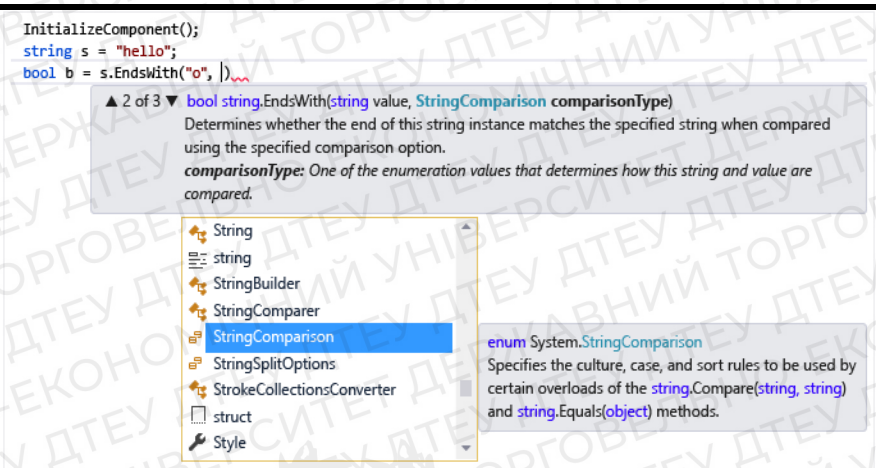

Рисунок 2.1 Загальний вигляд роботи IntelliSense

SQL Server Management Studio (SSMS) - це інтерактивне середовище для управління SQL Server. Воно дозволяє адміністраторам баз даних та розробникам працювати з базами даних SQL Server. SSMS надає зручний та простий інтерфейс для створення та настроювання баз даних, а також виконання SQL-запитів. За допомогою SSMS можна виконувати такі дії [16]:

— Створення та редагування баз даних: SSMS надає зручний інтерфейс для створення та редагування баз даних. Користувач може створювати нові бази даних, таблиці, зберігати процедури, функції та інші об'єкти баз даних.

— Виконання SQL-запитів: SSMS дозволяє виконувати SQL-запити для отримання потрібних даних з баз даних. Запит можна ввести вручну або скористатися вбудованими засобами SSMS для створення складних запитів.

— Налаштування та моніторинг баз даних: SSMS надає засоби для моніторингу та налаштування баз даних. Користувач може настроювати параметри баз даних, змінювати їх конфігурацію та моніторити роботу баз даних.

— Резервне копіювання та відновлення баз даних: SSMS надає засоби для резервного копіювання та відновлення баз даних. Користувач може створювати резервні копії баз даних, щоб забезпечити захист від втрати даних в разі аварійного завершення роботи сервера або інших проблем.

— Управління безпекою баз даних: SSMS надає інструменти для управління безпекою баз даних. Користувач може настроювати права доступу до таблиць та інших об'єктів баз даних, а також налаштовувати інші параметри безпеки.

— Аналіз та оптимізація запитів: SSMS дозволяє аналізувати запити, що виконуються на базі даних, та оптимізувати їх роботу. Користувач може використовувати засоби SSMS для побудови плану виконання запитів, виявлення проблем в роботі бази даних та їх усунення.

— Інтеграція з іншими інструментами: SSMS може бути інтегрований з іншими інструментами Microsoft, такими як Visual Studio або Excel. Це дозволяє користувачеві зручно працювати з даними з різних джерел та використовувати їх для аналізу та звітності.

Усі ці можливості роблять SQL Server та SSMS потужними інструментами для управління та аналізу даних. Вони використовуються в багатьох компаніях та організаціях по всьому світу для забезпечення ефективної та безпечної роботи з даними.

2.2 Моделювання та аналіз програмного забезпечення

Загальні процеси, які проходить користувач, включають у себе вибір потрібного пункту меню, отримання інформації у візуальному форматі, редагування таблиць за допомогою інтерфейсу користувача.

Опис редагування таблиць та отримання інформації:

— Користувач обирає пункт меню Інгредієнти, додає інгредієнт, використовуючи спеціальну форму. Програма автоматично додає його до бази даних та змінює візуальне відображення цієї та інших сторінок.

— Користувач обирає пункт меню Залишки, для перегляду актуальної інформації про наявність та кількість товарів на складі.

— Користувач обирає пункт Постачання чи Списання, відповідно до обраного пункту, він може додати постачання чи зробити списання товарів через спеціальну форму. Програма автоматично додасть їх у базу даних та внесе зміни у інші таблиці, а також, у візуальну частину інших сторінок програми.

— Користувач обирає пункт Страви у меню, в цьому пункті він може додати страви, які складаються з певних Інгредієнтів, певної кількості. Додавання відбувається через спеціальну форму.

2.3 Архітектура програмного забезпечення

Для розробки додатку було використано паттерн розробки MVVM. MVVM (Model-View-ViewModel) - це архітектурний паттерн програмування, який використовується для побудови користувацького інтерфейсу додатків [17].

У MVVM інтерфейс додатку розбивається на три частини:

— Model - це частина додатку, яка відповідає за роботу з даними та бізнес-логікою. Це може бути, наприклад, база даних або веб-сервіс.

— View - це частина додатку, яка відповідає за відображення даних та обробку користувацького вводу. Це може бути, наприклад, графічний інтерфейс користувача (GUI) або сторінка веб-сайту.

— ViewModel - це проміжна логіка між Model та View. ViewModel отримує дані від Model та форматує їх у спосіб, який може бути

відображений на View. ViewModel також обробляє вхідні дані від View та передає їх до Model для збереження.

Основна ідея MVVM полягає в тому, щоб View та Model були максимально незалежні одне від одного. View не повинен мати прямого доступу до Model, а Model не повинен знати про існування View. ViewModel ж розташований між ними та координує їх роботу [17].

MVVM дозволяє підвищити тестованість додатку, оскільки ViewModel може бути легко протестований, адже він не залежить від View або Model. Крім того, цей паттерн дозволяє зменшити зв'язок між компонентами додатку та підвищити його розширюваність [17].

На рисунку 2.1 показана модель роботи паттерну MVVM.

Рис. 2.1 Модель роботи паттерну MVVM

Для написання додатку було вирішено використовувати мову програмування C# для написання функціональних модулів, а для проектування інтерфейсу користувача використовується Windows Presentation Foundation.

2.3.1 Опис архітектури програмного забезпечення

Програма містить в собі 3 основні частини: View, Model та View-Model. Кожна з частин відповідає за певну частину функціоналу. В таблиці 2.1 описані кожна з частин та класи, що до нього входять.

Таблиця 2.1

Опис View, Model, View-Model та відповідних класів

Назва частини	Клас	Опис та призначення класу
1	2	3
Model	RelayCommand	Реалізовує інтерфес ICommand, необхідний для роботи команд, що пов'язують View та Model через частину View-Model
	StoreHouseContext	Реалізує необхідний контекст даних, для зв'язку з базою даних
	DbUsage	Клас, який містить основні методи, необхідні для взаємодії з базою даних
	AddToDb	Реалізує механізм додавання даних у таблиці бази даних
	DeleteFromDb	Реалізує механізм видалення даних з таблиць бази даних
	ChangeDb	Реалізує механізм зміни даних у таблицях бази даних
	Dish, Ingredient, Ingredient, WriteOff	Описують модель бази даних, їх зв'язки між собою
View	DishesUC, IngredientsUC, MainUC, RemainsUC, SuppliesUC, WriteOffsUC	Відповідають за запуск та відображення візуальної частини кожної конкретної сторінки додатку
	MainWindow	Відповідає за запуск візуальної частини головної сторінки додатку
	AddIngredientWindow	Відповідає за відображення форми додавання інгредієнтів
View-Model	DishesUCViewModel, IngredientsUCViewModel, MainUCViewModel, MainWindowsViewModel, MenuViewModel, RemainsUCViewModel	Дані класи відповідають за обробку даних з частини View та передачу їх до частини Model

Продовження таблиці 2.1

1	2	3
	AddIngredientViewModel	Відповідає за обробку даних з частини View, вікна додавання інгредієнтів та передачу їх до частини Model

2.3.2. Розробка та проектування архітектури бази даних

В результаті розробки програмного додатку було спроектовано базу даних. Проектування бази даних зазвичай починається з аналізу вимог до даних, які будуть зберігатися в базі даних. Це може включати визначення потрібних таблиць, полів, відносин між таблицями та інших аспектів.

Основні кроки проектування бази даних:

— Визначення сутностей та атрибутів: Визначте сутності, які будуть представлені в базі даних, і їх властивості (атрибути). Наприклад, якщо проектується база даних для інтернет-магазину, можуть бути сутності "клієнти", "замовлення", "товари" та інші.

— Визначення відносин між сутностями: Визначте відносини між сутностями, такі як один до одного (1:1), один до багатьох (1:N) або багато до багатьох (N:M). Наприклад, у базі даних інтернет-магазину зв'язок між сутностями "клієнти" і "замовлення" може бути один до багатьох, тоді як зв'язок між сутностями "замовлення" і "товари" буде багато до багатьох.

— Нормалізація даних: Забезпечте, щоб дані у базі даних були нормалізовані. Це означає, що кожна таблиця містить тільки унікальні дані, а повторювані дані зберігаються у окремих таблицях. Це забезпечує ефективність, цілісність та консистентність даних.

— Визначення ключів: Визначте ключі для кожної таблиці, щоб забезпечити унікальність записів. Ключ може бути одним або декількома полями.

— Визначення типів даних: Визначте тип даних для кожного поля у таблиці. Наприклад, текстові поля можуть бути визначені як VARCHAR, числові поля як INT або FLOAT, дати як DATE або DATETIME тощо.

Після визначення необхідних даних, були спроектовані логічна та фізична моделі бази даних

Логічна модель бази даних (LDM) описує структуру даних та взаємозв'язки між ними відносно бізнес-логіки додатка. LDM допомагає проектувальникам баз даних визначити таблиці, відносини та атрибути даних, які будуть використовуватися у додатку. LDM може бути використана для документування вимог до бази даних та є основою для розробки фізичної моделі бази даних.

Фізична модель бази даних (PDM) описує, як дані будуть зберігатися та оброблятися в конкретній СУБД (системі управління базами даних). PDM включає фізичні структури, такі як таблиці, індекси, обмеження, типи даних тощо. PDM зазвичай створюється на основі LDM, але деталізує специфічні деталі СУБД.

Розроблені логічна та фізична моделі бази даних представлені на рисунках 2.2 та 2.3 нижче.

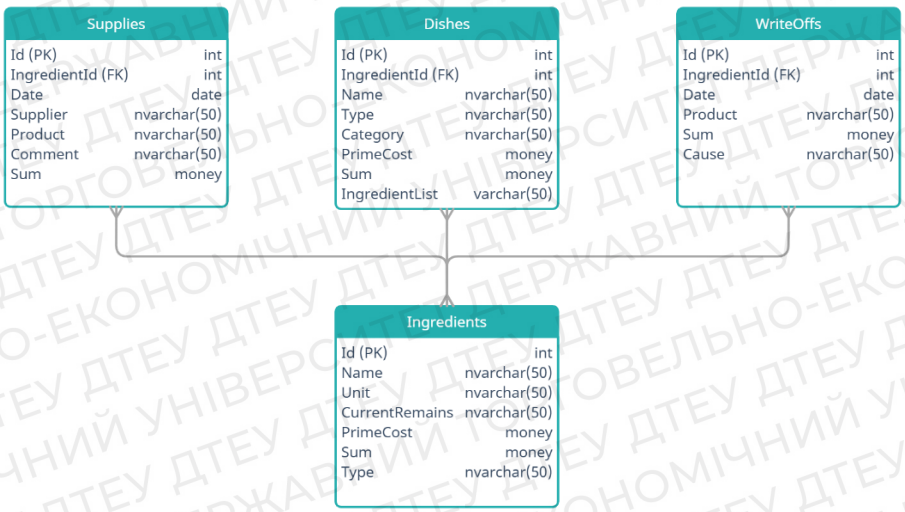

Рис. 2.2. Фізична модель бази даних

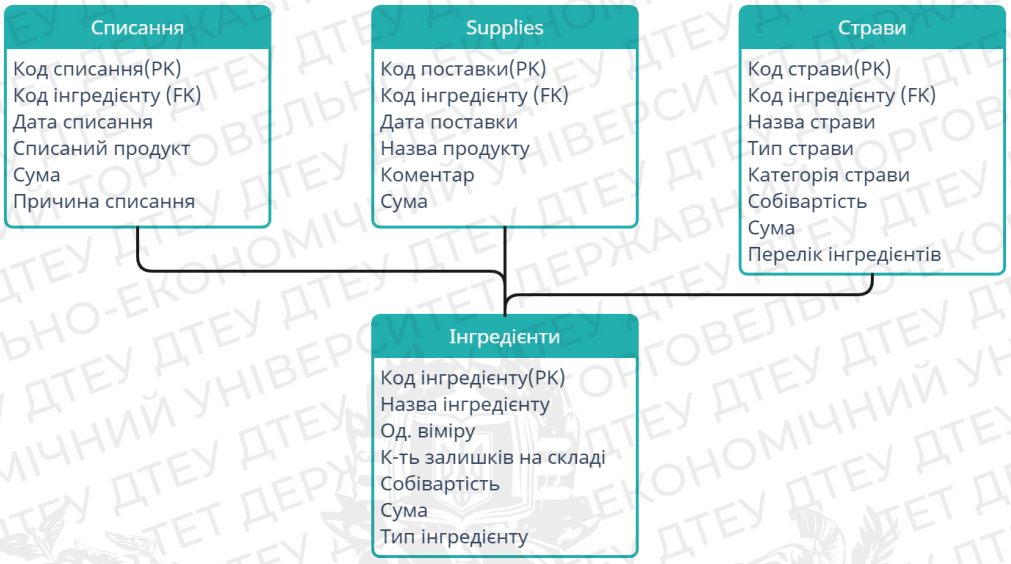

Рис. 2.3 Логічна модель бази даних

2.3.3 Розробка та проектування архітектури проекту

В результаті проектування архітектури проекту, було створено діаграму класів.

Діаграма класів - це графічне представлення структури класів та їх взаємозв'язків у системі. Вона використовується для опису архітектури програмного забезпечення та допомагає програмістам зрозуміти, як класи взаємодіють один з одним та які властивості та методи мають кожен клас.

Діаграма класів складається з класів, що зображаються у вигляді прямокутників, та взаємозв'язків між цими класами, що зображаються у вигляді стрілок. Кожен клас містить список його властивостей та методів, а також ім'я класу.

Взаємозв'язки між класами можуть бути різних типів, наприклад, агрегація, композиція, спадкування, залежність. Кожен тип взаємозв'язку має свій вигляд та відображає відношення між класами.

Діаграми класів використовуються в проектуванні програмного забезпечення, для опису взаємодії між класами, і для покращення зрозуміння структури системи як для розробників, так і для інших зацікавлених сторін, таких як менеджери проекту, клієнти, тестувальники та інші.

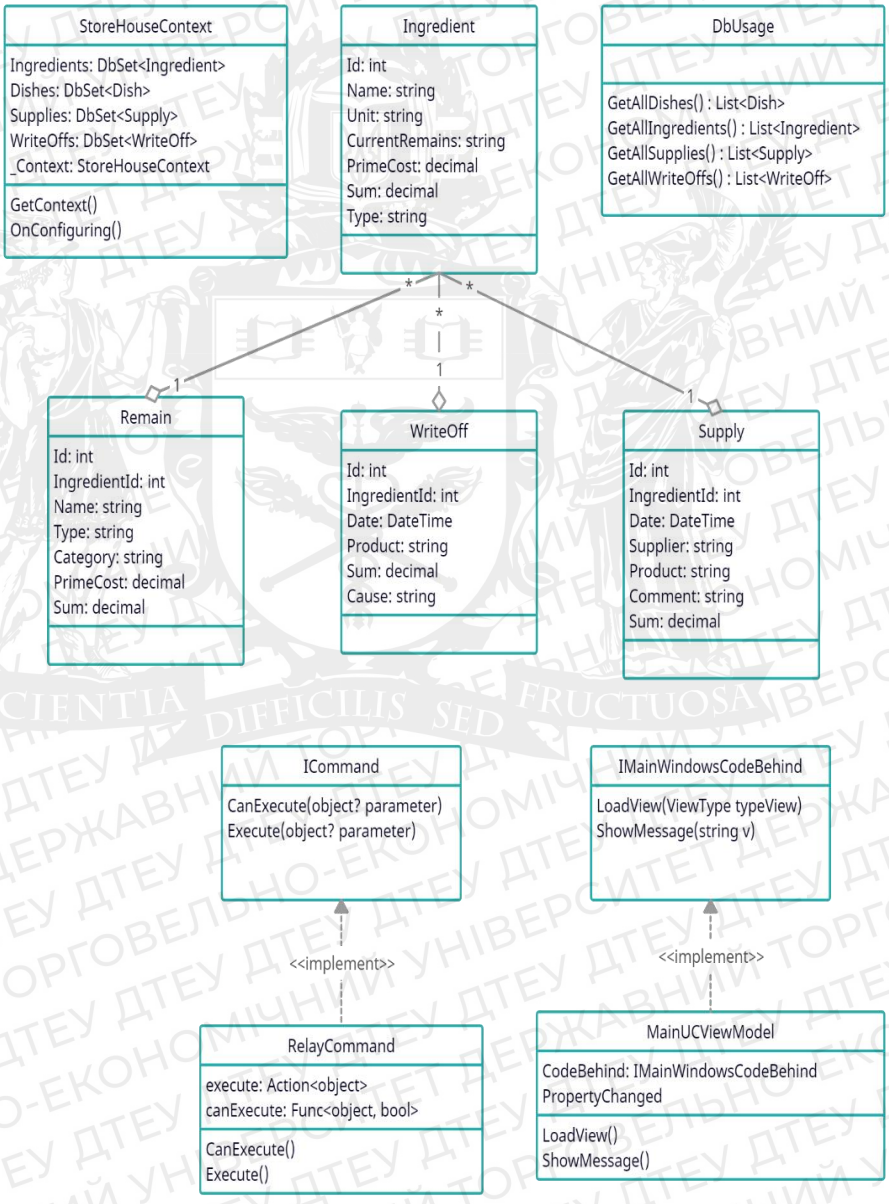

Рис. 2.4. Діаграма класів

2.4. Висновки до розділу

В результаті розробки архітектури програмного забезпечення, було розроблено архітектуру бази даних. Була створена фізична модель бази даних, яка описує, як дані будуть зберігатися та оброблятися в конкретній СУБД. Фізична модель включає фізичні структури, такі як таблиці, індекси, обмеження, типи даних. Також була створена логічна модель бази даних, яка описує структуру даних та взаємозв'язки між ними відносно бізнес-логіки додатка. Логічна модель допомогла нам визначити таблиці, відносини та атрибути даних, які будуть використовуватися у додатку.

Також було розроблено діаграму класів - графічне представлення структури класів та їх взаємозв'язків у системі. Ми описали взаємодію між класами, та розробили архітектурну структуру програмного забезпечення.

РОЗДІЛ 3
ОПИС ЗАПРОПОНОВАНОГО ТЕХНІЧНОГО РІШЕННЯ

3.1 Процес розробки програмного забезпечення

Оскільки для розробки програмного забезпечення нами було обрано паттерн MVVM, першим етапом буде створення візуального представлення програмного забезпечення. Розробка користувацького інтерфейсу була реалізована можливостями вбудованої мови розмітки XAML. Основними тегами, які використовувались при написанні коду були: StackPanel, Border, Grid, Image, Button. На рисунку нижче наведено приклад коду (рис. 3.1) та його візуальне відображення (рис. 3.2).

```
<StackPanel Grid.Row="0">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="50"></ColumnDefinition>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Image Grid.Column="0" VerticalAlignment="Top" Margin="0,5,0,0" Source="../Images/Logo(Склад).png" Width="25" Height="25"></Image>
        <Expander Padding="5" Grid.Column="1" Style="{DynamicResource ResourceKey=ExpanderStyle1}"  FontSize="16" FontWeight="SemiBold" Header="Склад">
            <StackPanel >
                <Button Command="{Binding Path=(???).LoadRemainsUCCommand}" Padding="5" FontWeight="Regular" HorizontalContentAlignment="Left" BorderThi
                <Button Command="{Binding Path=(???).LoadSuppliesUCCommand, UpdateSourceTrigger=PropertyChanged}" Padding="5" FontWeight="Regular" Horiz
                <Button Command="{Binding Path=(???).LoadWriteOffsUCCommand, UpdateSourceTrigger=PropertyChanged}" Padding="5" FontWeight="Regular" Hori
            </StackPanel>

        </Expander>
    </Grid>
</StackPanel>
<StackPanel Grid.Row="1">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="50"></ColumnDefinition>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Image Grid.Column="0" VerticalAlignment="Top" Margin="0,5,0,0" Source="../Images/Logo (Меню).png" Width="25" Height="25"></Image>
        <Expander Padding="5" Grid.Column="1" Style="{DynamicResource ResourceKey=ExpanderStyle1}"  FontSize="16" FontWeight="SemiBold" Header="Меню">
            <StackPanel>
                <Button Command="{Binding Path=(???).LoadDishesUCCommand, UpdateSourceTrigger=PropertyChanged}" Padding="5" FontWeight="Regular" Horizor
                <Button Command="{Binding Path=(???).LoadIngredientsUCCommand, UpdateSourceTrigger=PropertyChanged}" Padding="5" FontWeight="Regular" Ho
            </StackPanel>

        </Expander>
    </Grid>
</StackPanel>
```

Рис. 3.1 Приклад коду візуального відображення

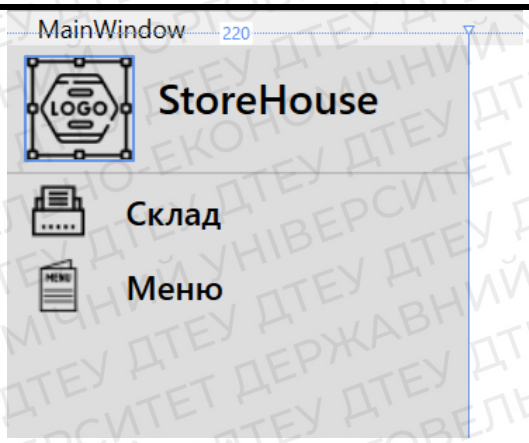

Рис. 3.2 Візуальне відображення коду XAML

Таким чином, було створено головну сторінку користувацького інтерфейсу, шість сторінок відображення даних, а також шістнадцять допоміжних сторінок. Нижче наведена структура візуальної частини програмного забезпечення (Рис. 3.3).

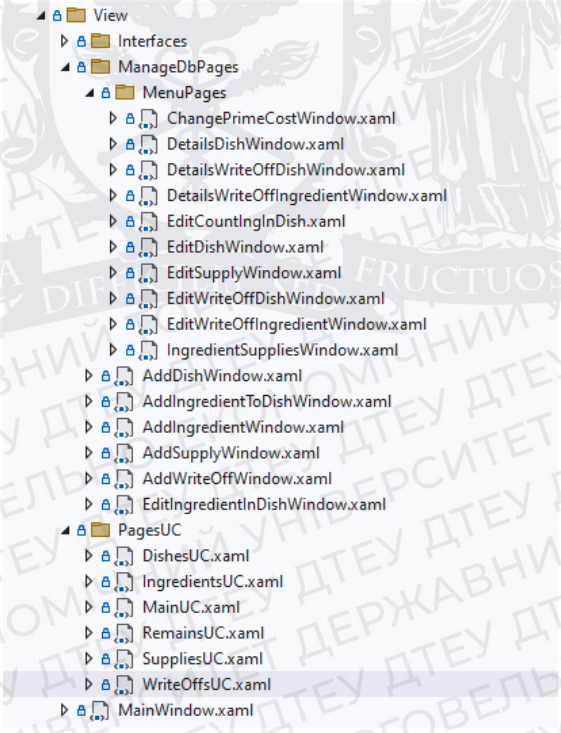

Рис. 3.3 Структура візуальної частини програмного забезпечення

Наступним кроком реалізації проекту за паттерном MVVM є створення моделі програмного забезпечення. Оскільки дані для роботи програми ми отримуємо з бази даних, модель повинна відповідати її структурі. Таким чином, кожен окремий клас основної моделі повинен відповідати таблиці у базі даних, а поля у цих класах – їх стовбцям. Нижче наведено приклад реалізації класу моделі (Рис. 3.4).

```
Ссылок: 13
public class Ingredient
{
    Ссылок: 11
    public int Id { get; set; }
    Ссылок: 12
    public string Name { get; set; }
    Ссылок: 6
    public string Unit { get; set; }
    Ссылок: 48
    public string CurrentRemains { get; set; }
    Ссылок: 20
    public decimal PrimeCost { get; set; }
    Ссылок: 18
    public decimal Sum { get; set; }
    Ссылок: 3
    public string Type { get; set; }


}
```

Рис. 3.4 Реалізація класу моделі

Для зв’язку з базою даних було вирішено використовувати Entity Framework Core. Для коректної роботи бібліотеки, ми реалізували StoreHouseContext (Рис. 3.5), у якому була визначена основна конфігурація підключення до бази даних.

```
Ссылок: 99+
class StoreHouseContext : Microsoft.EntityFrameworkCore.DbContext
{

    Ссылок: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(connectionString: @"Server=DESKTOP-O510GJE;Database=StoreHouse;Trusted_Connection=True; TrustServerCertificate=True");
    }

    Ссылок: 22
    public DbSet<Ingredient> Ingredients { get; set; }
    Ссылок: 12
    public DbSet<Dish> Dishes { get; set; }
    Ссылок: 9
    public DbSet<Supply> Supplies { get; set; }
    Ссылок: 13
    public DbSet<WriteOff> WriteOffs { get; set; }
    Ссылок: 12
    public DbSet<OutputAddDish> OutputAddDishes { get; set; }


    private static StoreHouseContext _Context;
    Ссылок: 4
    public static StoreHouseContext GetContext()
    {
        _Context = new StoreHouseContext();
        return _Context;
        //if (_Context == null)
        //    _Context = new StoreHouseContext();
        //return _Context;
    }
}
```

Рис. 3.5 Код класу StoreHouseContext

Для зміни даних у базі даних було реалізовано такі класи, як: AddToDb, DeleteFromDb, EditDb. У цих класах було реалізовано методи, які звертаються до бази даних та змінюють дані у таблицях. Приклад коду в класі AddToDb наведено нижче (Рис. 3.6). Для витягування даних з бази даних та для роботи з ними було реалізовано відповідні методи у класі DbUsage (Рис. 3.7).

```
Ссылок: 14
internal class AddToDb : IAddToDb
{
    Ссылок: 3
    public string AddOutputAddDish(int dishId, string name, string count, string sum)
    {
        string result = "Готово!";
        using (StoreHouseContext db = new StoreHouseContext())
        {
            OutputAddDish outputAddDish = new OutputAddDish()
            {
                DishId = dishId,
                Name = name,
                Count = count,
                Sum = sum

            };
            db.OutputAddDishes.Add(outputAddDish);
            db.SaveChanges();
        }

        return result;
    }
```

Рис. 3.6 Приклад коду в класі AddToDb

```
Ссылок: 2
public static List<OutputIngredient> SearchIngredientsByName(string name)
{
    try
    {
        List<Ingredient> IngList;
        using (StoreHouseContext db = new StoreHouseContext())
        {
            IngList = (from ingredient in db.Ingredients
                where ingredient.Name.Contains(name)
                select ingredient).ToList();
        }
        List<OutputIngredient> dataList = new List<OutputIngredient>();
        foreach (var temp Ingredient in IngList)
        {
            var tempRemains string[] = temp.CurrentRemains.Split( separator: ' ');
            dataList.Add( item: new OutputIngredient(temp.Name, temp.Unit,
                currentRemains: $"{Convert.ToString(Math.Round(Convert.ToDecimal(tempRemains[0].Replace( oldChar: '.', newChar: ',')), 2))}{temp.Unit}",
                primeCost: $"{Math.Round(temp.PrimeCost, 2)}грн", sum: $"{Math.Round(temp.Sum, 2)}грн", temp.Type));
        }
        return dataList;
    }
    catch (SqlNullValueException e)
    {
        return new List<OutputIngredient>();
    }
}
```

Рис. 3.7 Метод класу DbUsage

Останньою частиною імплементації паттерну MVVM є розробка частини ViewModel, що пов'язує модель програми та її візуальне відображення. Було створено двадцять три класи, що реалізують зв'язок класів моделі та файлів XAML, а також основну логіку роботи програми. Нижче наведено приклад класу ViewModel (Рис. 3.8).

```
Ссылок: 3
class AddIngredientToDishViewModel : INotifyPropertyChanged
{
    private IMainWindowsCodeBehind _MainCodeBehind;

    Ссылок: 1
    public AddIngredientToDishViewModel(IMainWindowsCodeBehind codeBehind)
    {
        if (codeBehind == null) throw new ArgumentNullException(nameof(codeBehind));

        _MainCodeBehind = codeBehind;
    }


    //Fields
    private static string _Count;
    Ссылок: 2
    public string Count
    {
        get => _Count;
        set
        {
            _Count = value;
            Sum = Math.Round(DbUsage.GetSum(DbUsage.GetPrimeCost(DbUsage.GetIngredientIdByName(SeletedProduct), SeletedProduct), currentRemains: _Count),2)
            OnPropertyChanged();
        }
    }
```

Рис. 3.8 Частина коду класу AddIngredientToDishViewModel

Основний принцип передачі даних у користувацький інтерфейс полягає у використанні класів ViewModel у ролі посередника. Керуючись правилами паттерну MVVM, усі зв'язки досягаються в наслідок

використання команд. Для зручної роботи з ними було створено клас RelayCommand (Рис. 3.9), в якому описана основна конфігурація, необхідна для використання команд. Екземпляри даного класу зазвичай використовується у класах ViewModel (Рис. 3.10). Після об'явлення команди, ми маємо підв'язати її до коду у представленні через відповідну властивість (Рис. 3.11).

```
Ссылок: 99+
internal class RelayCommand : ICommand
{
    private Action<object> execute;
    private Func<object, bool> canExecute;


    public event EventHandler CanExecuteChanged
    {
        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    Ссылок: 75
    public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
    {
        this.execute = execute;
        this.canExecute = canExecute;
    }


    Ссылок: 0
    public bool CanExecute(object parameter)
    {
        return this.canExecute == null || this.canExecute(parameter);
    }

    Ссылок: 0
    public void Execute(object parameter)
    {
        this.execute(parameter);
    }
}
```

Рис. 3.9 Клас RelayCommand

```
<Button Command="{Binding Path=(???).LoadRemainsUCCommand}" I
```

Рис. 3.11 Властивість Command у файлі XAML

```
private RelayCommand _ChangePrimeCost;
Ссылок: 0
public RelayCommand ChangePrimeCost
{
    get
    {
        return _ChangePrimeCost ?? new RelayCommand(execute: obj :object =>
        {
            EditDb edit = new EditDb();
            {
                edit.EditIngredientPrimeCost(
                    DbUsage.GetIngredientIdByName(GetChoosenIngredientItem().Name),
                    NewPrimeCost
                    );
            }
            _MainCodeBehind.LoadView(ViewType.Ingredients);
        });
    }
}
```

Рис. 3.10 Приклад використання команди у класі IngredientsUCViewModel

3.2 Інструкція користувача та огляд програмного забезпечення

Для розгортання програмного забезпечення необхідно мати комп'ютер чи ноутбук із встановленою операційною системою Windows 10 або наступних

версій. Також мають бути встановлені такі програми, як Microsoft SQLServer версії 2019 року або наступних для роботи з базою даних. Для запуску програми необхідно запустити файл StoreHouse.exe.

При запуску програми, користувач потрапляє на головку сторінку (Рис. 3.12). З лівого боку є доступ до меню, через яке здійснюється навігація за сторінками програми.

Рис. 3.12 Головна сторінка програмного забезпечення

При натисканні на кнопку «Залишки», користувач потрапляє на сторінку з інформацією про поточні залишки на складі (Рис. 3.13). На сторінці дані відображені в виді таблиці, з автоматичним сортуванням за стовпцями зі стовпцями Назва, Тип, Залишки, Собівартість та Сума. Також на сторінці реалізоване поле для пошуку товарів за назвою. Знаходячись на цій сторінці, ми можемо подивитись усі постачання обраного продукту на склад. Відображення цих даних відбувається на інформаційній сторінці (Рис. 3.14), що включає у себе таблицю постачань зі стовбцями Номер, Дата, Постачальник, Товари, Коментар, Сума, а також кнопку повернення на попередню сторінку.

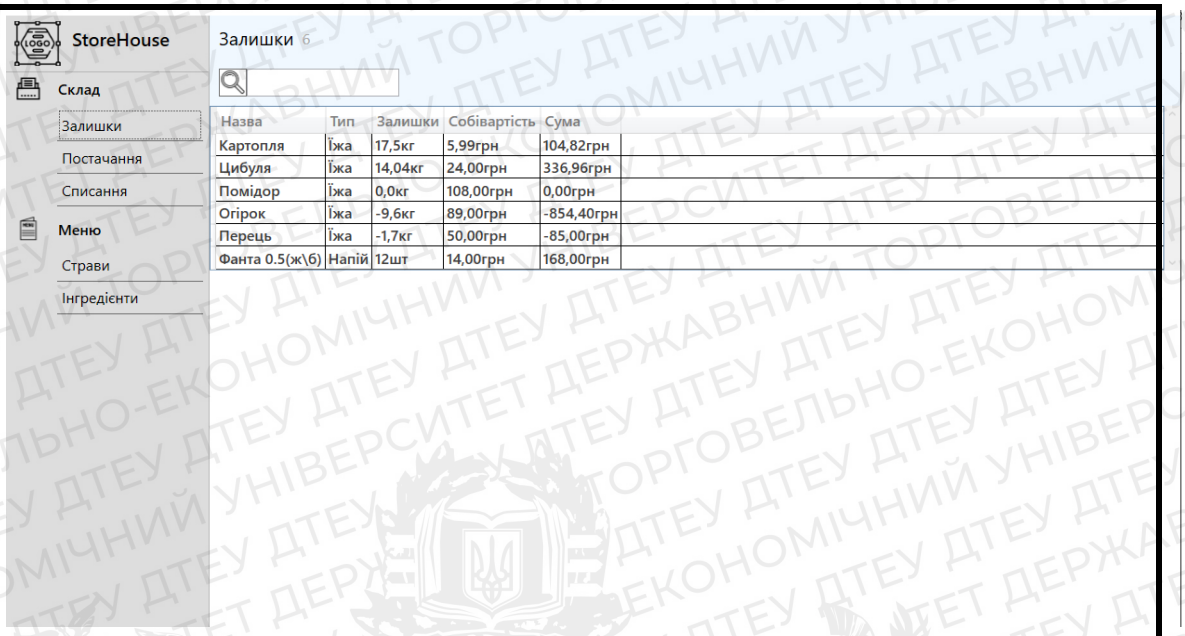

Рис. 3.13 Сторінка Залишки

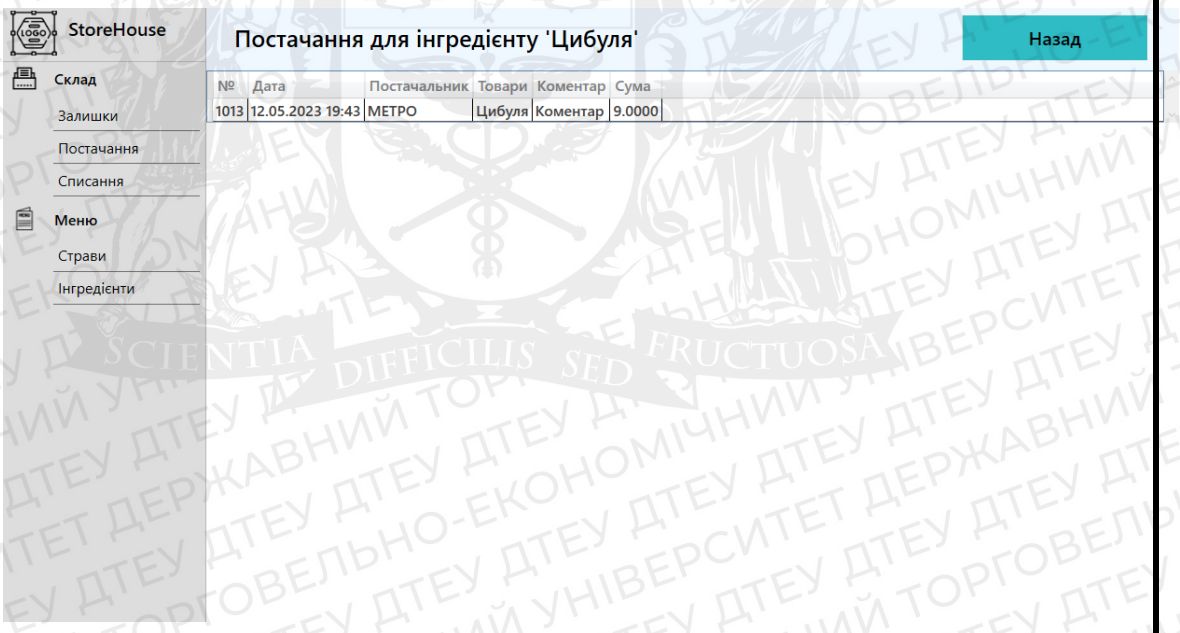

Рис. 3.14 Інформаційна сторінка постачань обраного товару

При натисканні на кнопку «Постачання», користувач потрапляє на сторінку з інформацією про постачання на склад (Рис. 3.15). На сторінці дані відображені в виді таблиці, з автоматичним сортуванням, зі стовпцями Номер, Дата, Постачальник, Товари, Кількість, Сума та Коментар. Також на

сторінці реалізоване поле для пошуку товарів за назвою. Знаходячись на цій сторінці, ми можемо додати постачання, редагувати або видалити його. Після натискання кнопки «Додати постачання» буде відкрито вікно форми (Рис. 3.16) з полями Найменування, Постачальник, Кількість, Собівартість, Загальна сума та Коментар. Після натискання кнопки «Редагувати» буде відкрито вікно форми (Рис. 3.17) з полями Найменування, Постачальник, Кількість, Собівартість, Загальна сума, а також кнопка для видалення обраного постачання.

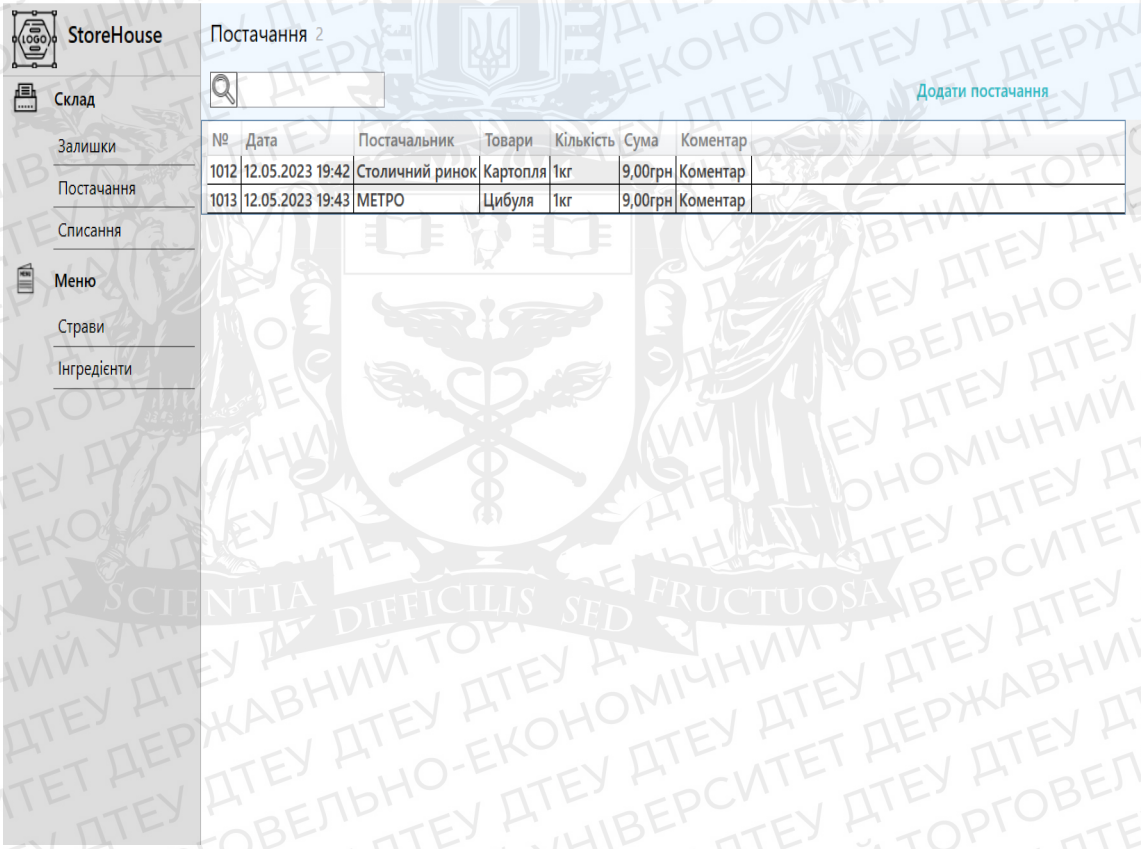

Рис. 3.15 Сторінка Постачання

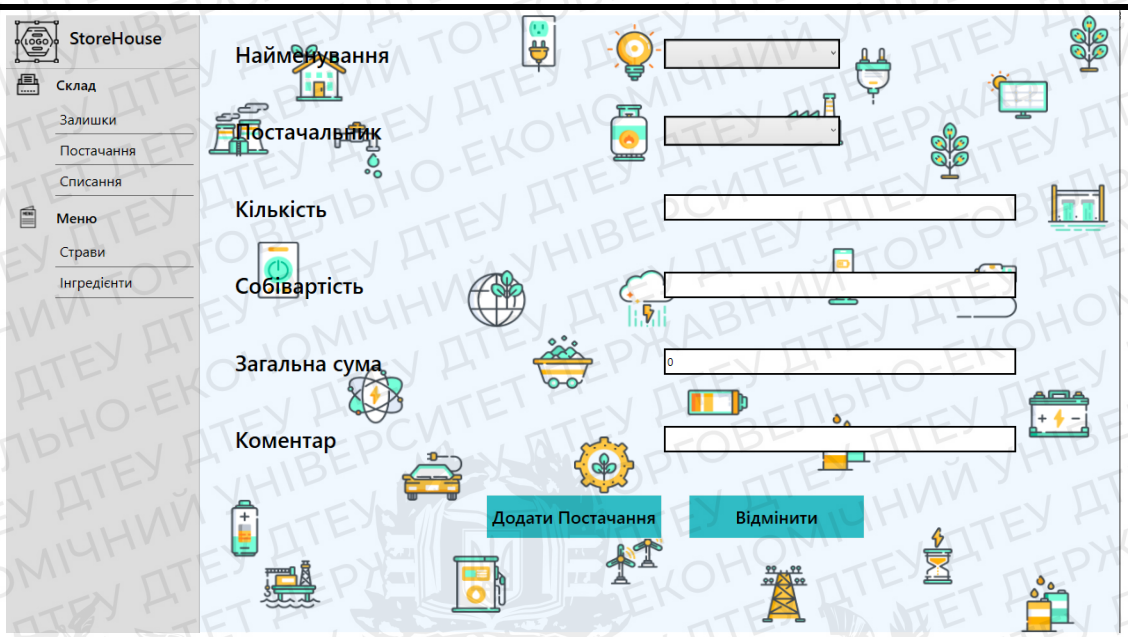

Рис. 3.16 Вікно форми Додавання постачання

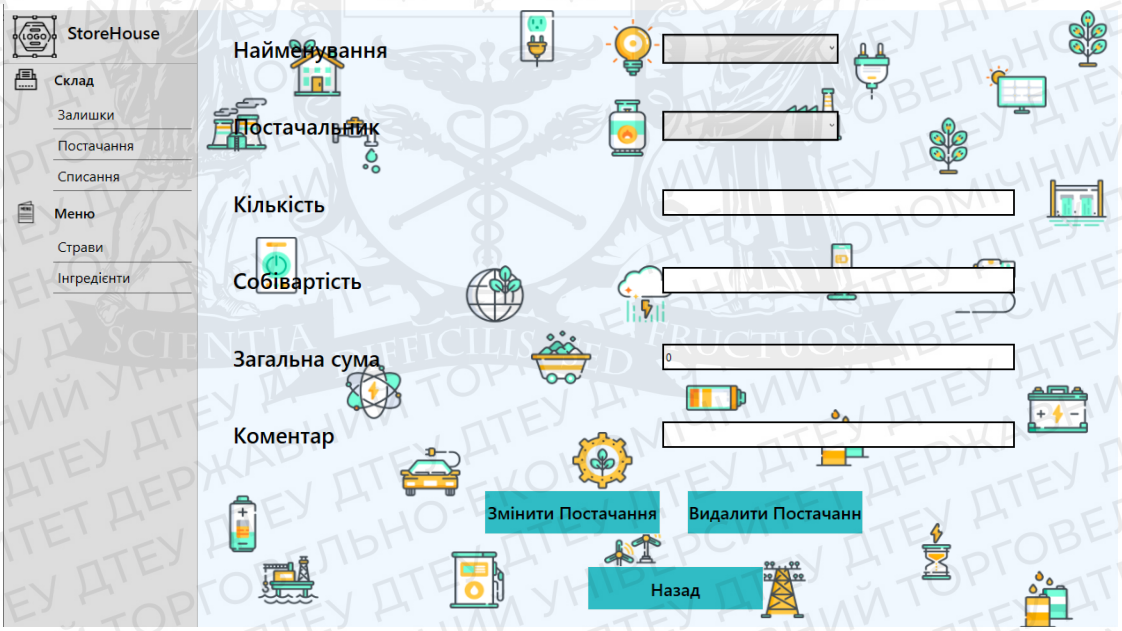

Рис. 3.17 Вікно форми Зміни та Видалення постачання

При натисканні на кнопку «Списання», користувач потрапляє на сторінку з інформацією про списання зі складу (Рис. 3.18). На сторінці дані відображені в виді таблиці, з автоматичним сортуванням, зі стовпцями Дата,

Товар, Кількість, Сума та Причина. Також на сторінці реалізоване поле для пошуку товарів за назвою. Знаходячись на цій сторінці, ми можемо додати списання, редагувати або видалити його. Після натискання кнопки «Додати списання» буде відкрито вікно форми (Рис. 3.19) з полями Найменування, Кількість, Коментар, Сума. Після натискання кнопки «Деталі інгредієнту» буде відкрито інформаційне вікно (Рис. 3.20), у якому описані усі списані інгредієнти. Після натискання кнопки «Редагувати» буде відкрито вікно форми (Рис. 3.21) з полями Найменування, Кількість, Коментар, Сума, а також кнопка для видалення обраного постачання.

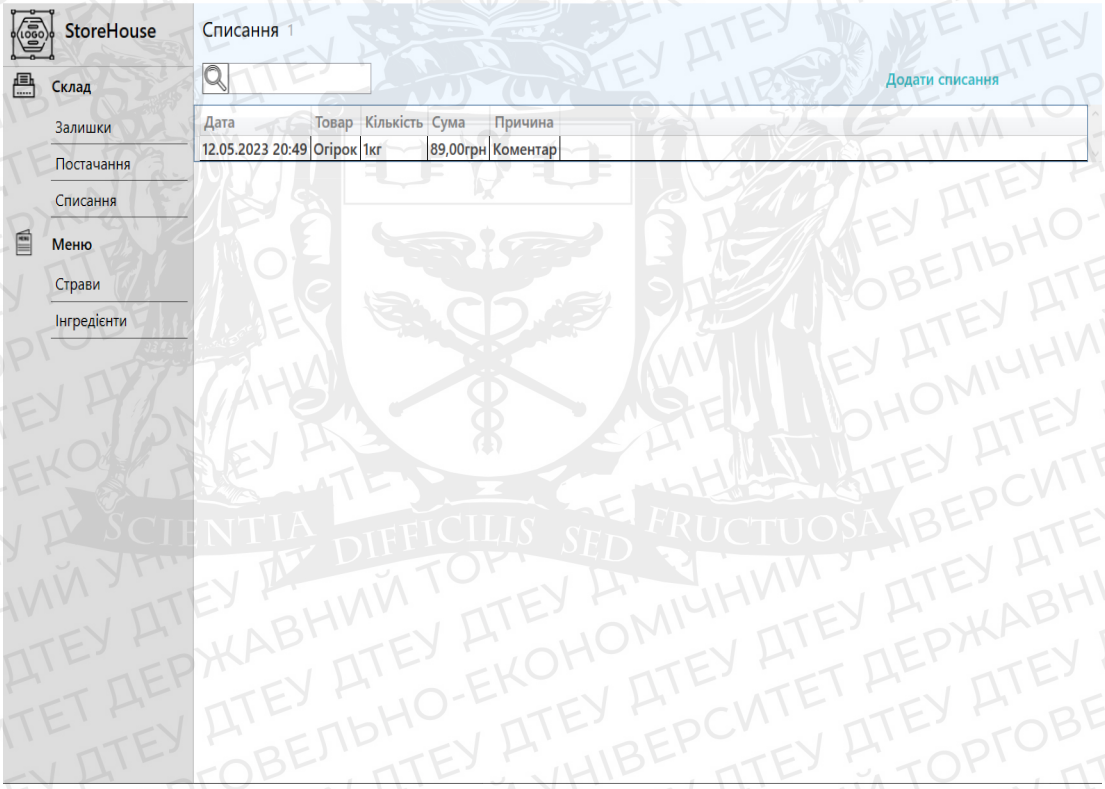

Рис. 3.18 Сторінка Списання

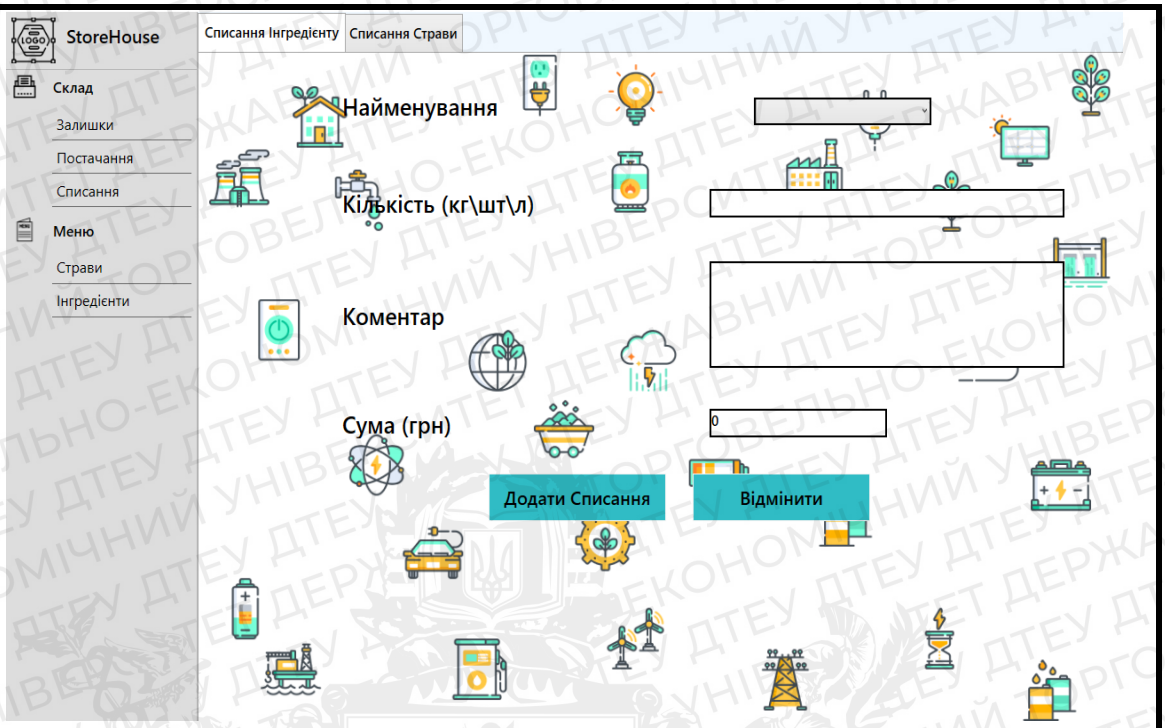

Рис. 3.19 Форма додавання списання

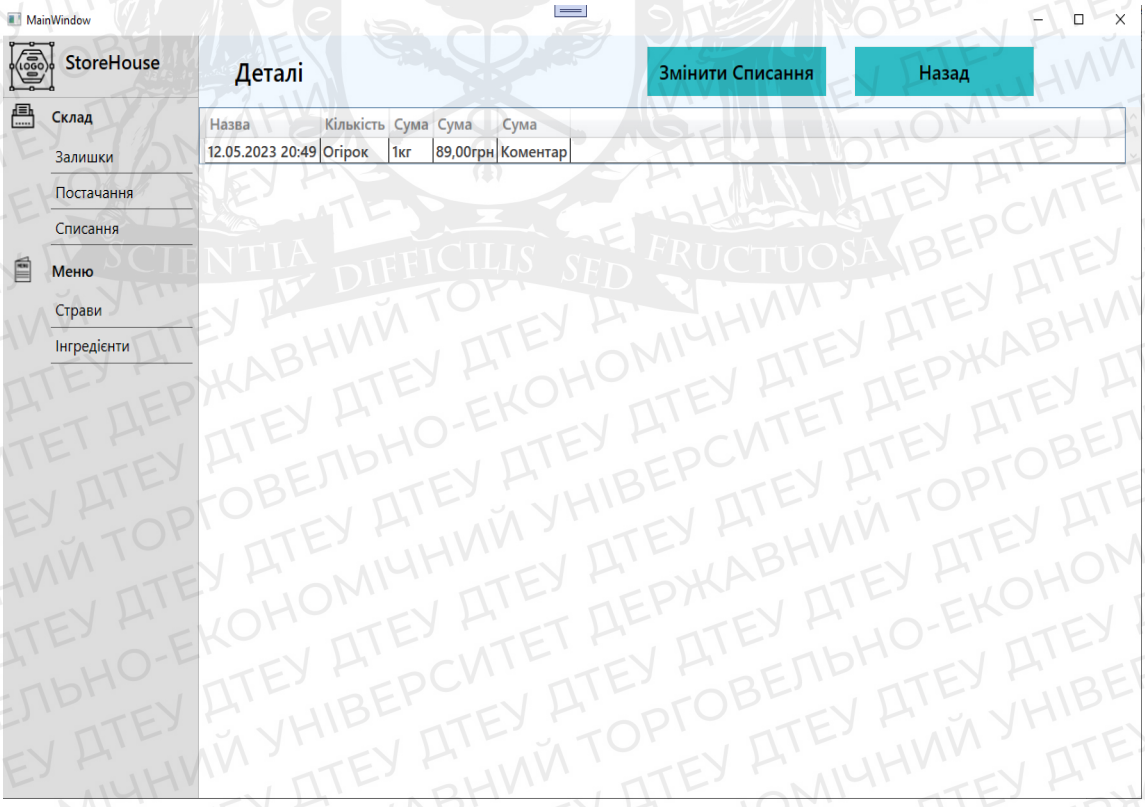

Рис. 3.20 Інформаційна сторінка списань обраного товару

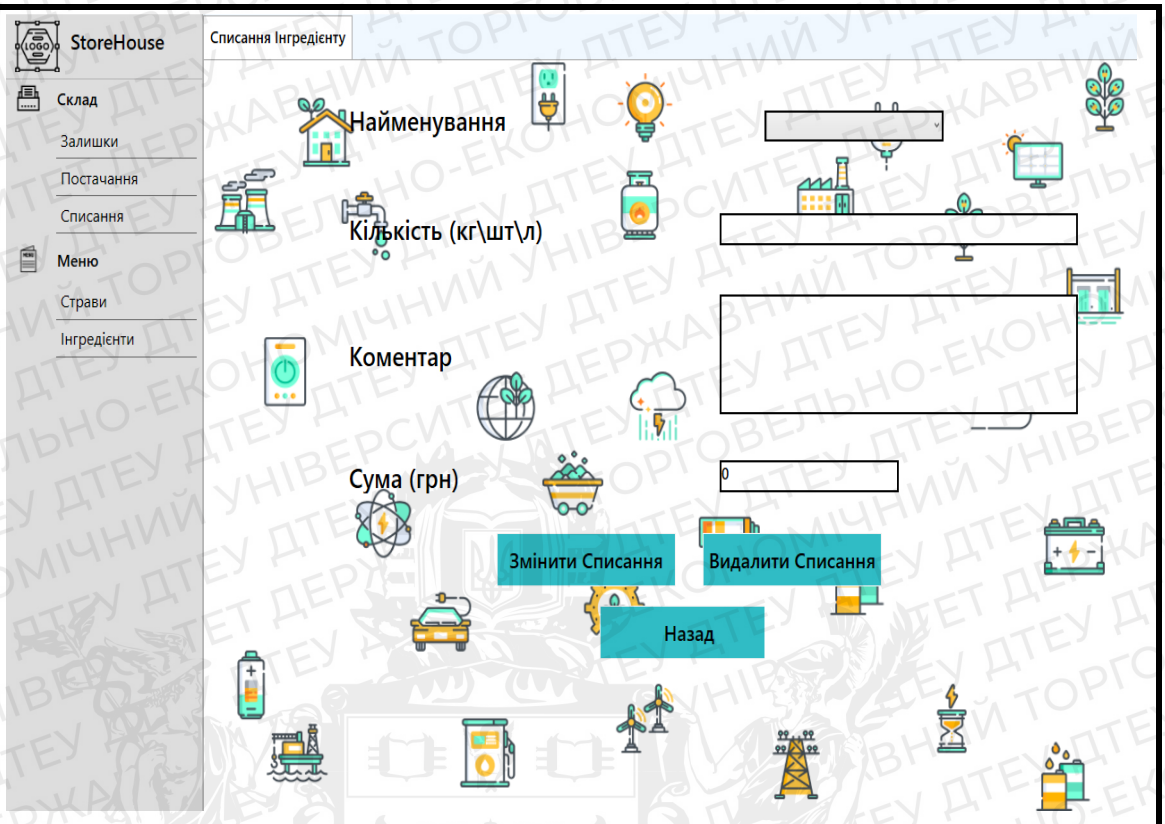

Рис. 3.21 Форма Зміни та Видалення списання

При натисканні на кнопку «Страви», користувач потрапляє на сторінку з інформацією про додані до меню страви (Рис. 3.22). На сторінці дані відображені в виді таблиці, з автоматичним сортуванням, зі стовпцями Назва, Тип, Категорія, Собівартість та Ціна. Також на сторінці реалізоване поле для пошуку страв за назвою. Після натискання кнопки «Додати страву» буде відкрито вікно форми (Рис. 3.23) з можливістю обрати список інгредієнтів, що сходять до страви, її тип, категорію, ціну, переглянути собівартість страви, поле вводу назви. Після натискання кнопки «Деталі» буде відкрито інформаційне вікно (Рис. 3.24) у якому описані усі інгредієнти, що входять до страви. Після натискання кнопки «Редагувати» буде відкрито вікно форми (Рис. 3.25) з можливістю редагування ціни, списку інгредієнтів, а також кнопка для видалення обраної страви.

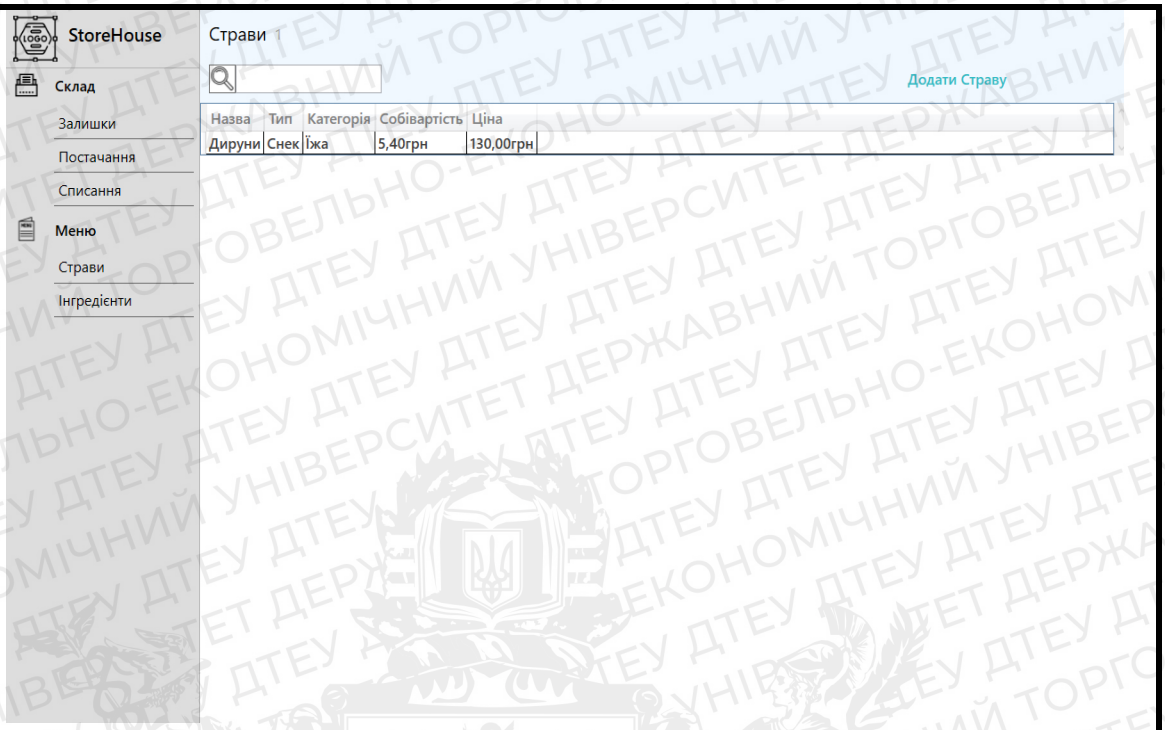

Рис. 3.22 Сторінка Страви

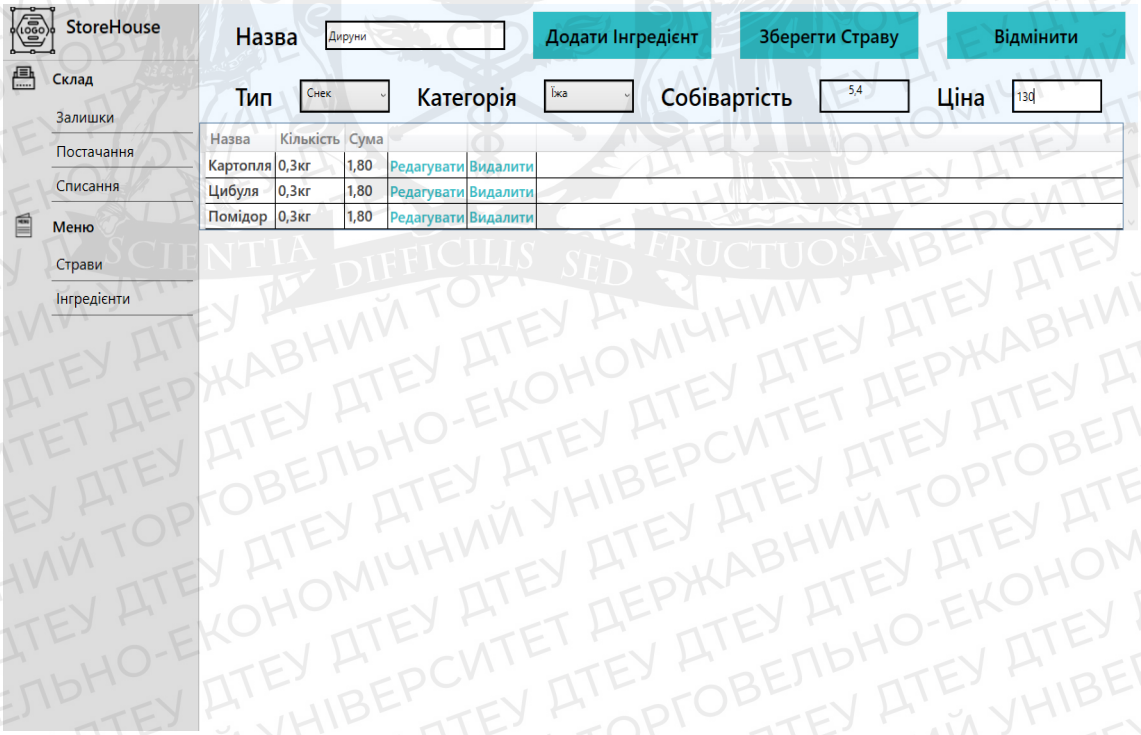

Рис. 3.23 Форма додавання страви

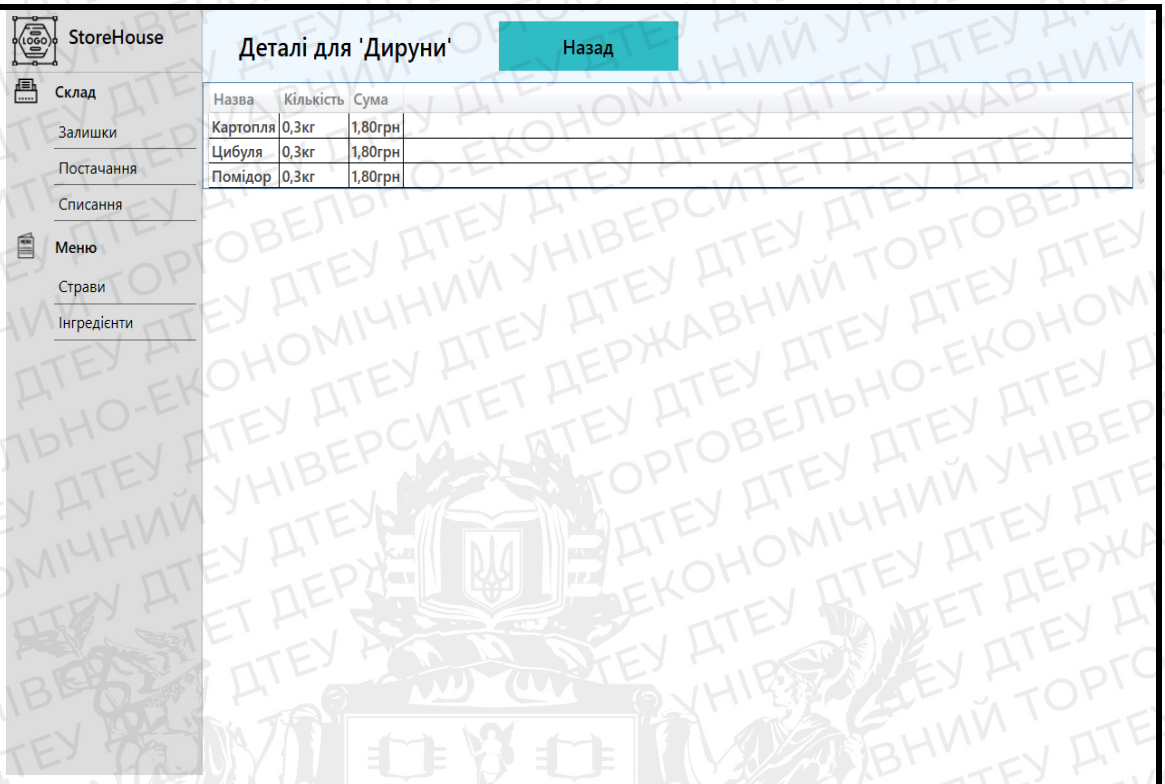

Рис. 3.24 Інформаційна сторінка інгредієнтів обраної страви

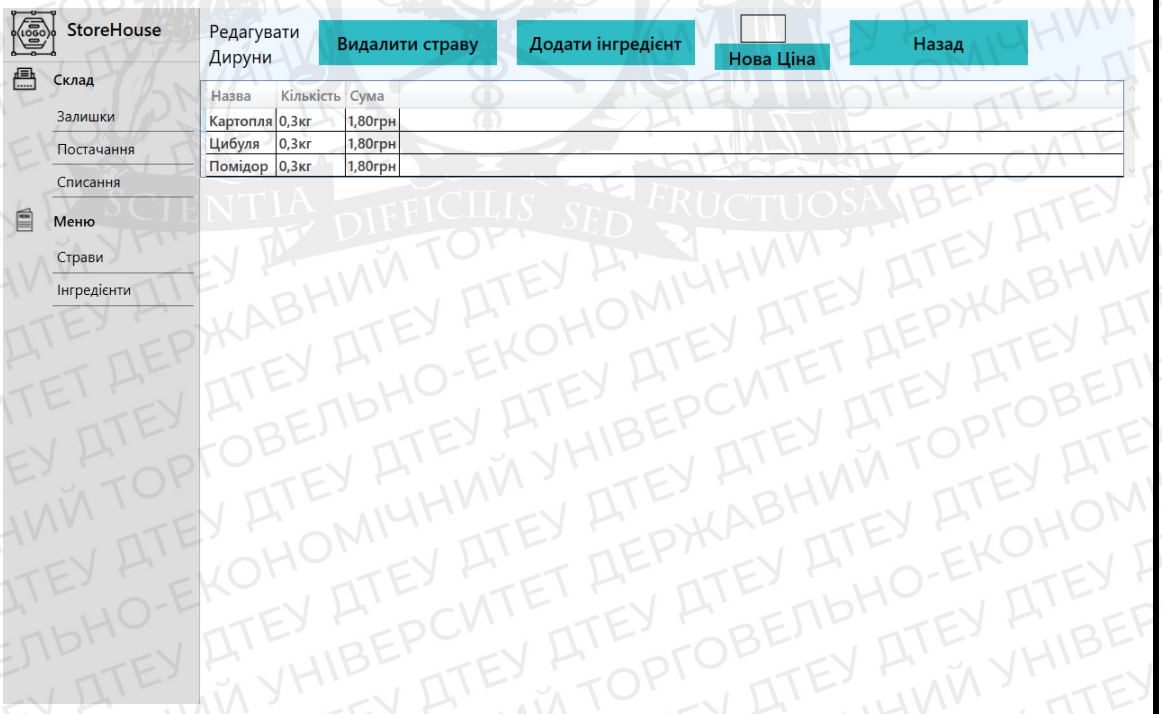

Рис. 3.25 Форма Зміни та Видалення страви

При натисканні на кнопку «Інгредієнти», користувач потрапляє на сторінку з інформацією про додані до складу інгредієнти (Рис. 3.26). На сторінці дані відображені в виді таблиці, з автоматичним сортуванням, зі стовпцями Назва, Од. Виміру, Залишки на складі, Собівартість та Сума залишку. Також на сторінці реалізоване поле для пошуку страв за назвою та можливість змінити собівартість інгредієнту. Після натискання кнопки «Додати інгредієнт» буде відкрито вікно форми (Рис. 3.27) з полями Назва, Одиниці виміру, Тип, Кількість залишків та Собівартість.

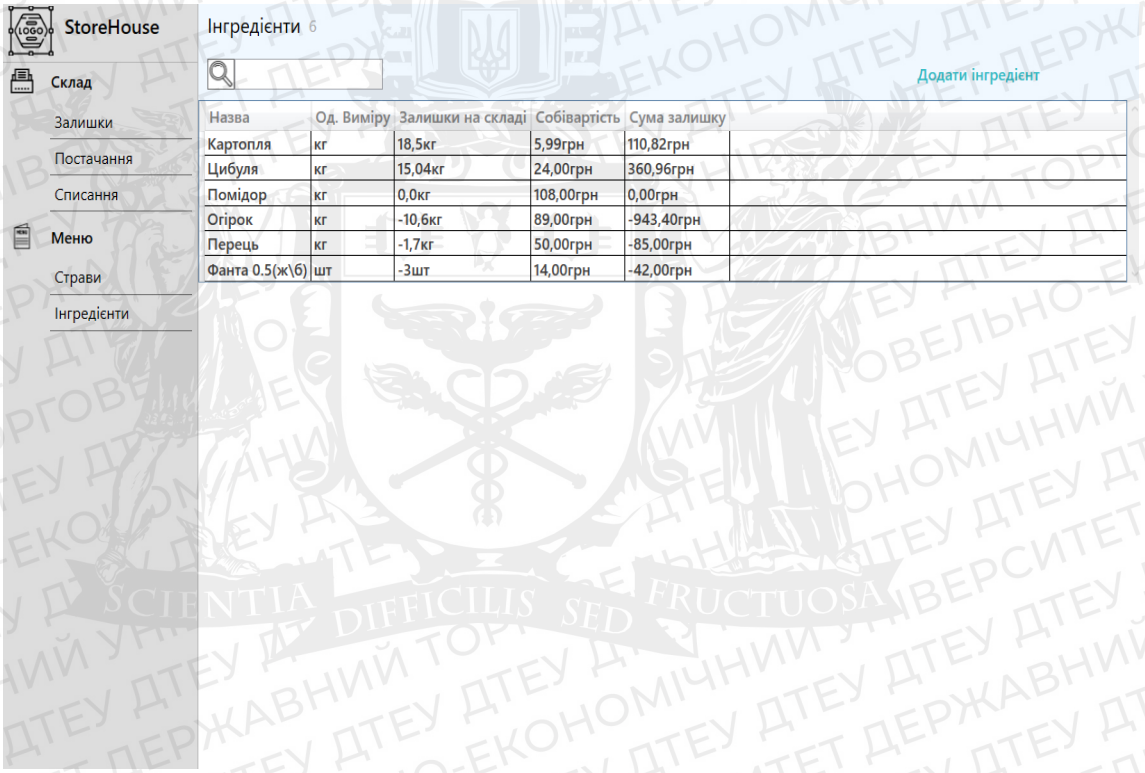

Рис. 3.26 Сторінка Інгредієнтів

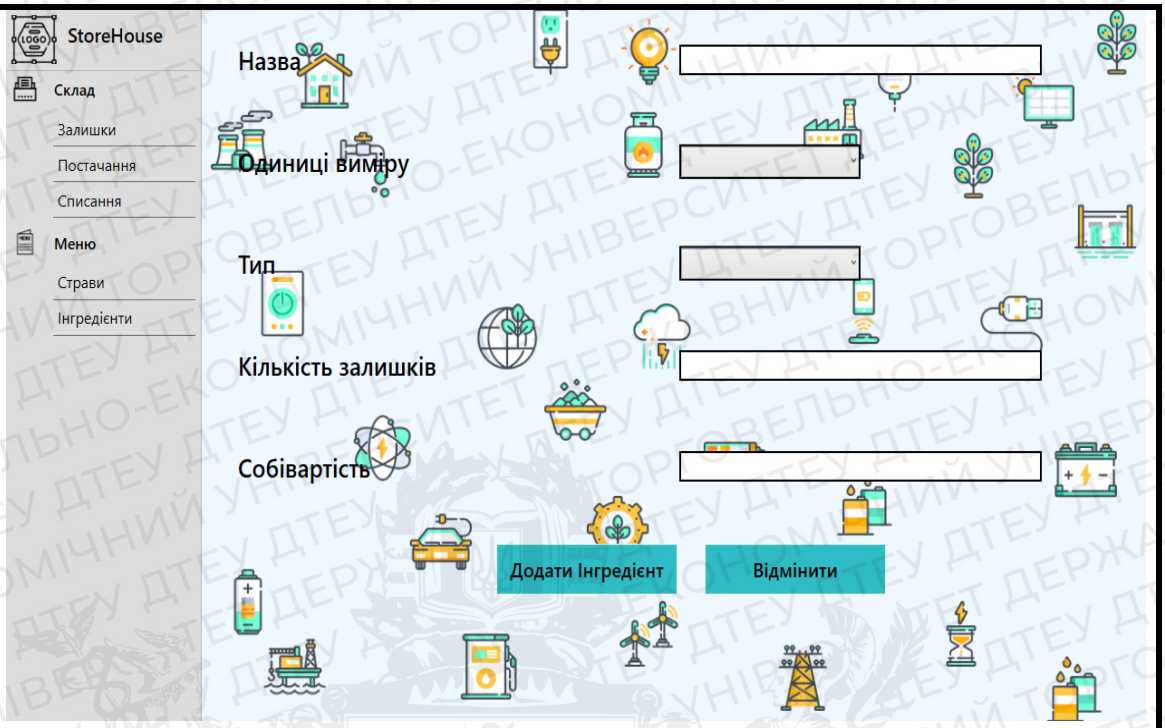

Рис. 3.27 Форма додавання інгредієнту

3.3 Висновки до розділу

У даному розділі було детально описано процес розробки програмного забезпечення, використовуючи паттерн розробки MVVM. Були створені 3 частини: View, Model, ViewModel.

Також було наведено детальний опис програмного забезпечення та інструкції користувача. Були наведені ілюстровані приклади використання Додатку а також опис кожної його сторінки.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Дипломний проект присвячений розробці програмного модулю управління товарами у ресторані.

Тема є актуальною, адже управління товарами у ресторані має велику практичну значущість, оскільки може допомогти ресторанам збільшити продуктивність та ефективність управління своїми бізнес-процесами.

У дипломній роботі було описано види існуючих POS систем, їх переваги та недоліки. Описано їх процес діяльності та їх значущість для ресторану. Також було проаналізовано відомі програмні продукти, їх переваги та недоліки. В результаті виявилось, що безкоштовних програмних засобів, що підходять для роботи з малим бізнесом та які підтримують українську мову не існує. Такий висновок робить створення програми на дану тему актуальною.

У пункті «Розробка технічного завдання та функціональних вимог» було наведено загальні положення та опис предметного середовища. Було розроблено нефункціональні вимоги та функціональні вимоги, такі як «Списання товару», «Додавання поставок» та «Перегляд залишків», варіанти використання, технічне завдання, головна мета, призначення та цілі розробки програмного забезпечення.

В пункті «Архітектура програмного забезпечення» була поетапно описана модель роботи користувача з програмою. Описаний принцип роботи такого архітектурного паттерну як MVVM.

Окрім цього була детально описана архітектура проекту. В результаті було спроектовано 3 модулі: View, Model та ViewModel. View - це частина додатку, яка відповідає за відображення даних та обробку користувацького

вводу. Model - це частина додатку, яка відповідає за роботу з даними та бізнес-логікою. ViewModel - це проміжна логіка між Model та View. ViewModel отримує дані від Model та форматує їх у спосіб, який може бути відображений на View. ViewModel також обробляє вхідні дані від View та передає їх до Model для збереження. Також були спроектовані та детально описані основні необхідні класи та утиліти забезпечення їх коректної роботи.

У пункті «Розробка та проектування архітектури баз даних» було спроектовано базу даних, за такими основними кроками як: Визначення сутностей та атрибутів, Визначення відносин між сутностями, Нормалізація даних, Визначення ключів та Визначення типів даних. Також було розроблено фізична модель бази даних, яка описує, як дані будуть зберігатися та оброблятися в конкретній СУБД та логічна модель бази даних, яка описує структуру даних та взаємозв'язки між ними відносно бізнес-логіки додатка. Логічна модель допомогла нам визначити таблиці, відносини та атрибути даних, які будуть використовуватися у додатку.

У пункті «Розробка та проектування архітектури проекту» була розроблена діаграма класів - графічне представлення структури класів та їх взаємозв'язків у системі. Ми описали взаємодію між класами, та розробили архітектурну структуру програмного забезпечення.

В розділі «Опис запропонованого технічного рішення» наведено детальний опис розгортання програми на комп’ютері користувача та вказані загальні етапи роботи з нею.

Перспективою подальшої розробки є перенесення бази даних у хмарне сховище, реалізація веб-інтерфейсу та налаштування комунікації зі сторонніми приладами виводу інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Restaurant POS systems: The most-asked questions, answered. SumUp - a better way to get paid. [Електронний ресурс]. – Режим доступу: https://www.sumup.com/en-ie/business-guide/restaurant-pos-most-asked-questions-answered/

2. Cloud vs On-premise POS software: Which one should be your business choice?. World's #1 POS for Magento. [Електронний ресурс]. – Режим доступу: https://www.magestore.com/blog/cloud-vs-on-premise-pos-software/

3. Hayes A. What Is a POS System and How Does It Work?. Investopedia. [Електронний ресурс]. – Режим доступу: https://www.investopedia.com/terms/p/point-of-sale.asp

4. Chen J. Android Operating System (OS): Definition and How It Works. Investopedia. [Електронний ресурс]. – Режим доступу: https://www.investopedia.com/terms/a/android-operating-system.asp

5. Bluetooth Protocol (Part 1): Basics and Working. Engineers Garage. [Електронний ресурс]. – Режим доступу: https://www.engineersgarage.com/bluetooth-protocol-part-1-basics-and-working/

6. CISSP-ISSAP T. B. Wireless Networking Protocols Explained. Lifewire. [Електронний ресурс]. – Режим доступу: https://www.lifewire.com/wireless-networking-protocols-explained-2486947

7. What is an Application Programming Interface (API)? | IBM. IBM - Deutschland | IBM. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/topics/api

8. Why you should integrate Toast POS with WISK inventory software. WISK Restaurant Software | 5x Faster Than Spreadsheets. [Електронний ресурс]. – Режим доступу: https://www.wisk.ai/blog/why-you-should-integrate-toast-pos-with-wisk-inventory-software-and-how-to-do-it

9. Lightspeed Restaurant POS Review [The Best POS for Restaurants]. TheRealBarman. [Електронний ресурс]. – Режим доступу: https://therealbarman.com/lightspeed-pos-review/

10. Loyverse Free POS Software review | Business Review. Business Review. [Електронний ресурс]. – Режим доступу: https://www.businessrevieweurope.eu/loyverse-free-pos/

11. Square Review 2023: Features, Pros & Cons. Forbes Advisor. [Електронний ресурс]. – Режим доступу: https://www.forbes.com/advisor/business/software/square-review/

12. Flachman T. Frequently Asked Questions about POS Systems. Bepoz. [Електронний ресурс]. – Режим доступу: https://bepoz.com/blog/frequently-asked-questions-pos-systems/

13. ReSharper: The Visual Studio Extension for .NET Developers by JetBrains. JetBrains. [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/resharper/

14. Introduction To C# Programming Using Visual Studio .Net Framework. Software Testing Help. [Електронний ресурс]. – Режим доступу: https://www.softwaretestinghelp.com/c-sharp/introduction-to-csharp/

15. A tour of C# - Overview. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

16. Transact-SQL reference (Database Engine) - SQL Server. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс]. –

Режим доступу: https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16

17. Contributor T. What is Model-View-ViewModel (MVVM)? | Definition from TechTarget. WhatIs.com. [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/whatis/definition/Model-View-ViewModel

ДОДАТКИ

Додаток А

Код представлення сторінки Залишків

```
<UserControl
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:StoreHouse.View"
      xmlns:b="http://schemas.microsoft.com/xaml/behaviors"
      xmlns:vm="clr-namespace:StoreHouse.ViewModels" x:Class="StoreHouse.View.RemainsUC"
      mc:Ignorable="d"
      d:DesignHeight="700" d:DesignWidth="1060"
      Background="AliceBlue">
    <UserControl.DataContext>
        <vm:RemainsUCViewModel/>
    </UserControl.DataContext>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="6.97*"/>
            <RowDefinition Height="10.76*"/>
            <RowDefinition Height="82.27*"/>
        </Grid.RowDefinitions>
        <StackPanel Margin="10" Orientation="Horizontal" Grid.Row="0">
            <TextBlock VerticalAlignment="Center" FontWeight="Regular" FontSize="20"><Run
Text="Залишки"/></TextBlock>
            <TextBlock Text="{Binding RemainsCount, UpdateSourceTrigger=PropertyChanged}" Margin="9,3,0,0"
VerticalAlignment="Center" Foreground="#ACACAC" FontSize="16"></TextBlock>
        </StackPanel>
        <StackPanel Margin="10" Orientation="Horizontal" Grid.Row="1">
            <Button Command="{Binding SearchButtonCommand}" Height="30" Width="30" Background="White">
                <Image Height="25" Width="25" Source="../../Images/Search image.png"/>
            </Button>
            <TextBox FontWeight="SemiBold" FontSize="16" VerticalContentAlignment="Center" Text="{Binding
SearchBar, UpdateSourceTrigger=PropertyChanged}" Height="30" Width="164"/>
            <!--<TextBlock TextAlignment="Left" VerticalAlignment="Center" Margin="50,0,0,0" Foreground="#30BCC5"
FontSize="16" FontWeight="Medium" IsEnabled="False"><Run Text="Тип"/></TextBlock>
            <TextBlock TextAlignment="Left" VerticalAlignment="Center" Margin="20,0,0,0" Foreground="#30BCC5"
FontSize="16" FontWeight="Medium" IsEnabled="False"><Run Text="Категорія"/></TextBlock>
            <TextBlock TextAlignment="Left" VerticalAlignment="Center" Margin="20,0,0,0" Foreground="#30BCC5"
FontSize="16" FontWeight="Medium" IsEnabled="False"><Run Text="Фільтр"/></TextBlock>-->
        </StackPanel>
        <ScrollViewer Grid.Row="2">
            <DataGrid ItemsSource="{Binding AllRemains, UpdateSourceTrigger=PropertyChanged}"
SelectedItem="{Binding ChosenRemainsItem, UpdateSourceTrigger=PropertyChanged}" Foreground="#818181"
FontSize="16" FontWeight="Medium" x:Name="RemainsDataGrid" AutoGenerateColumns="False"
IsReadOnly="True">
                <DataGrid.ContextMenu>
                    <ContextMenu>
                        <MenuItem BorderThickness="0" Background="Transparent" FontSize="16" FontWeight="Medium"
Foreground="#30BCC5" Header="Постачання" Command="{Binding
LoadIngredientSuppliesCommand}"></MenuItem>
                    </ContextMenu>
                </DataGrid.ContextMenu>
                <DataGrid.Columns>
                    <DataGridTextColumn Foreground="#3F3E3E" Binding="{Binding Name,
UpdateSourceTrigger=PropertyChanged}"  Header="Назва"/>
                    <DataGridTextColumn Foreground="#3F3E3E" Binding="{Binding Type,
UpdateSourceTrigger=PropertyChanged}"  Header="Тип"/>
```

```
            <DataGridTextColumn Foreground="#3F3E3E" Binding="{Binding CurrentRemains,
UpdateSourceTrigger=PropertyChanged}" Header="Залишки"/>
            <DataGridTextColumn Foreground="#3F3E3E" Binding="{Binding PrimeCost,
UpdateSourceTrigger=PropertyChanged}" Header="Собівартість"/>
            <DataGridTextColumn Foreground="#3F3E3E" Binding="{Binding Sum,
UpdateSourceTrigger=PropertyChanged}" Header="Сума"/>
        </DataGrid.Columns>
      </DataGrid>
    </ScrollViewer>
  </Grid>
</UserControl>
```

Додаток Б

Код класу RemainsViewModel

```
using StoreHouse.Model.Commands;
using StoreHouse.Model.DbContext;
using StoreHouse.Model.OutputDataModels;
using StoreHouse.ViewModels.Interfaces;
using StoreHouse.ViewModels.ViewSettingMethods;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace StoreHouse.ViewModels
{
  internal class RemainsUCViewModel : INotifyPropertyChanged
  {
    //Fields
    private IMainWindowsCodeBehind _MainCodeBehind;

    //ctor
    public RemainsUCViewModel(IMainWindowsCodeBehind codeBehind)
    {
      if (codeBehind == null) throw new ArgumentNullException(nameof(codeBehind));

      _MainCodeBehind = codeBehind;
    }

    public RemainsUCViewModel(){}


    private static string _SearchBar;
    public string SearchBar
    {
      get => _SearchBar;
      set
      {
        _SearchBar = value;
        OnPropertyChanged();
      }
    }
    // ChosenRemainsItem Field
    private static List<OutputIngredient> _AllRemains = ViewSettings.GetOutputIngredients();
    public List<OutputIngredient> AllRemains
    {
      get => _AllRemains;
      set
      {
        _AllRemains = value;
        OnPropertyChanged();
      }
    }
```

```
public static void SetAllRemains() => _AllRemains = ViewSettings.GetOutputIngredients();
private static OutputIngredient _ChosenRemainsItem;
public OutputIngredient ChosenRemainsItem
{
    get => _ChosenRemainsItem;
    set
    {
        _ChosenRemainsItem = value;
        OnPropertyChanged();
    }
}
private static string _RemainsCount = Convert.ToString(DbUsage.GetAllIngredients().Count);
public string RemainsCount
{
    get => _RemainsCount;
    set
    {
        _RemainsCount = value;
        OnPropertyChanged();
    }
}

public static void SetRemainsCount() => _RemainsCount = Convert.ToString(DbUsage.GetAllIngredients().Count);
public static string GetChosenRemainsItemName()
{
    return _ChosenRemainsItem.Name;
}
//Commands

private RelayCommand _LoadIngredientSuppliesCommand;
public RelayCommand LoadIngredientSuppliesCommand
{
    get
    {
        return _LoadIngredientSuppliesCommand ?? new RelayCommand(obj =>
        {
            _MainCodeBehind.LoadView(ViewType.IngredientSupply);
        });
    }
}

private RelayCommand _SearchButtonCommand;
public RelayCommand SearchButtonCommand
{
    get
    {
        return _SearchButtonCommand ?? new RelayCommand(obj =>
        {
            AllRemains = DbUsage.SearchIngredientsByName(SearchBar);
        });
    }
}

private RelayCommand _ShowMessageCommand;
public RelayCommand ShowMessageCommand
{
    get
    {
        return _ShowMessageCommand ?? new RelayCommand(obj =>
        {
            _MainCodeBehind.ShowMessage("RemainsUC");
        });
    }
}
```

```
        #region INotifyPropertyChanged Implementation
        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
        #endregion
    }
}
```

Додаток В

Посилання на код GitHub

Посилання на весь код GitHub: https://github.com/SnieFox/StoreHouse .

ДОДАТКИ

Додаток А

Посилання на код GitHub

Посилання на весь код GitHub: https://github.com/SnieFox/StoreHouse .я