

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»



Кафедра інформаційних систем та технологій

Курсова робота

З дисципліни «Програмування – 2. Структури даних та алгоритми»

Тема: «Автоматизована система управління закладом харчування»

Керівник:

ст. викл. Хмелюк М.С.

«Допущена до захисту»

(особистий підпис керівника)

« » _____ 2022р.

Виконавець:

ст. Павлова Софія Олегівна

залікова книжка № ІС-1224

(особистий підпис виконавця)

« » _____ 2022р.

Захищений з оцінкою

(оцінка)

Члени комісії:

(особистий підпис)

(особистий підпис)

(розшифровка підпису)

(розшифровка підпису)

Київ – 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Кафедра ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Дисципліна: «Основи програмування»

Курс: 1. Група: ІС-12. Семестр: 2

ЗАВДАННЯ

на курсову роботу студента

Павлова Софія Олегівна

1. Тема роботи: Автоматизована система управління закладом харчування.

2. Строк здачі студентом закінченої роботи: 23.05.2022.

3. Вихідні дані до роботи:

Принципи програмування: ООП, фреймворк — Microsoft .NET Framework, інтерфейс програмування додатків — Windows Forms API, мова програмування — C#, середовище розробки — Visual Studio 2019, система контролю версій Git.

4. Зміст розрахунково – пояснювальної записки (перелік питань, що підлягають розробці):

Вступ; аналіз існуючих рішень; постановка задачі: основні вимоги до системи, моделювання сутностей, їх атрибутів та зв'язків між ними, опис сценаріїв виконання застосунку; реалізація основних компонентів застосунку: реалізація зв'язків багато до одного, реалізація зв'язків багато до багатьох, реалізація статичних методів, користувацький інтерфейс та діалоги користувача, перспективи покращення застосунку; висновки; перелік використаних джерел.

Додатки:

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Діаграма класів, діаграма прецедентів, діаграма представлення зв'язку *.*.

6. Дата видачі завдання: 06.04.2022.

КАЛЕНДАРНИЙ ПЛАН

№, п/п	Назва етапів виконання курсової роботи	Строк виконання етапів роботи	Підписи або примітки
1.	Ознайомлення з темою	06.04.2022	
2.	Постановка задачі, формування сутностей та зв'язків між ними	13.04.2022	
3.	Реалізація базового класу та наслідування	15.04.2022	
4.	Реалізація користувацького інтерфейсу	18.04.2022	
5.	Реалізація зв'язків багато до одного	18.04.2022	
6.	Реалізація зв'язку багато до багатьох	19.04.2022	
7.	Реалізація додавання/редагування/видалення сутностей	20.04.2022	
8.	Реалізація статичних методів	21.04.2022	
9.	Тестування програми на помилки	03.05.2022	
10.	Оформлення теоретичної частини звіту	04.05.2022	
11.	Опис основних функціональних складових програми	17.05.2022	
12.	Захист курсової роботи	23.05.2022	

Студент _____

(підпис)

(Софія ПАВЛОВА)

Керівник _____

(підпис)

Марина ХМЕЛЮК

(Софія ПАВЛОВА)

« ____ » _____ 20 ____ р.

АНОТАЦІЯ

Павлова С.О. Автоматизована система управління закладом харчування. КІП ім. Ігоря Сікорського, Київ, 2022.

Робота містить 32 с. тексту, 21 рисунок, 7 таблиць, посилання на 8 літературних джерел, додатки.

Ключові слова: Автоматизація, система управління, заклад харчування, об'єктно-орієнтоване програмування, наслідування, поліморфізм, статичні методи, .NET, Windows Forms.

Курсову роботу присвячено створенню застосунку автоматизованої системи адміністрування закладами харчування на прикладі кафе. Проаналізовано аналогів таких систем і визначено їх головні особливості. На основі цих особливостей висунуто основні функціональні вимоги до запропонованої моделі такої системи.

Даний застосунок був розроблений для отримання практичних навичок з об'єктно-орієнтованого програмування на мові програмування C#, а саме: реалізації наслідування, поліморфізму, побудові зв'язків багато до одного та багато до багатьох, реалізації статичних методів у класах. Цю роботу було виконано в рамках курсу ознайомлення з інтерфейсом програмування додатків Windows Forms API.

Змодельовані основні сутності системи, їх методи та атрибути, логічно обґрунтовано зв'язки між ними. Визначено головні сценарії виконання програми.

Розроблено програмне забезпечення для прийому гостьових замовлень та редагування основного меню. Створено робочу модель запропонованого застосунку системи. Передбачено можливість для масштабування розробленої моделі системи.

Отримані результати можуть бути корисними при автоматизації аналогічних чи подібних об'єктів.

ЗМІСТ

ВСТУП.....	6
1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	7
2 ПОСТАНОВКА ЗАДАЧІ.....	9
2.1 Основні вимоги до системи.....	9
2.2 Моделювання сутностей, їх атрибутів та зв'язків між ними	9
2.3 Опис сценарії виконання застосунку	15
3 РЕАЛІЗАЦІЯ ОСНОВНИХ КОМПОНЕНТІВ ЗАСТОСУНКУ	16
3.1 Структура проекту	16
3.2 Реалізація зв'язків багато до одного	17
3.3 Реалізація зв'язків багато до багатьох	19
3.4 Реалізація статичних методів.....	23
3.5 Користувацький інтерфейс та діалоги користувача.....	25
3.6 Перспективи покращення застосунку.....	29
ВИСНОВКИ.....	31
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	32

ВСТУП

Завдяки швидкому розвитку ресторанної індустрії, активно зростає й інтерес до автоматизованого управління, в міру посилення конкуренції в бізнесі. Обов'язкові очікування щодо витрат на автоматизацію в бізнес-бюджетах нових закладів харчування – є найліпшим доказом цього. Керівники ресторанного бізнесу розглядають використання автоматизованих систем як низку можливостей для збільшення прибутку, зниження операційних витрат і підвищення лояльності клієнтів, тим самим забезпечуючи довгострокову конкурентну перевагу на ринку попиту.

Саме тому дана курсова робота присвячена розробці автоматизованої системи управління закладом харчування.

Об'єктом розробки є система управління закладом харчування.

Предметом роботи є віконний застосунок для адміністрування закладом харчування.

Метою роботи є отримання практичних навичок з об'єктно-орієнтованого програмування на мові програмування C#.

Головною задачею – створення робочої моделі автоматизованої системи адміністрування рестораном, і реалізація в програмі основних принципів об'єктно-орієнтованого програмування (далі ООП) на мові програмування C#.

У рамках цієї роботи описано процес створення віконного застосунку для операційних систем Windows в інтерфейсі програмування додатків, що відповідає за графічний інтерфейс користувача – Windows Forms API.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Ресторанний бізнес в Україні активно розвивається. На сьогоднішній день автоматизація ресторанів та кафе 1 стала необхідною умовою для забезпечення конкурентоспроможності та швидкого розвитку бізнесу. Автоматизація ресторанного бізнесу надає реальну допомогу у виконанні багатьох функцій, виконуваних різними співробітниками підприємства. Дослідження 2 показали, що такі системи забезпечують автоматизацію інтегрованих облікових, виробничих, маркетингових та управлінських процесів.

Основними завданнями технології автоматизованого управління є:

- структурований облік продажів;
- контроль діяльності підприємства в цілому та персоналу, зокрема;
- централізоване управління меню і прейскурантом цін;
- отримання аналітичної звітності;
- можливість обробляти значні обсяги зовнішньої та внутрішньої інформації для прийняття ефективних управлінських рішень.

Крім того, упровадження таких технологій має багато переваг, які покращують якість ведення бізнесу:

- підвищення якості та швидкості обслуговування клієнтів;
- зменшення людського чиннику, що призводить до виникнення помилок при роботі з клієнтами;

На сьогодні відомо про деякі заклади харчування, які повністю перейшли на автономне управління. Вони використовують такі програмні комплекси, як: «Парус-Ресторан», АС «Рестарт», «1С: Підприємство 8. Ресторан».

Розберемо детальніше особливості автоматизованих програмних комплексів на прикладі системи АС «Рестарт» 3.

Даний комплекс призначений для автоматизації роботи касира, офіціанта, бармена, буфетника та адміністратора (дивись рисунок 1.1). Система використовує платформу MS Windows і формат бази даних MS SQL.

Кафе "Тричел" 25.02.2011 14:43:32 **Рестарт**

открыт / 4 Администратор

Чек №:
 ФР: 1С-Рарус: Учебный ФР №1
 Карта:

	Наименование	Кол-во	Цена	Скидка	Сумма
1	Стейк из форели северных морей	1	700.00	0.00	700.00
2	Рибай с зеленым салатом	1	0.00	0.00	0.00
3	Стейк из молодого бычка	1	600.00	0.00	600.00
4	Мильфей с клубникой и кремом "Сабайон"	1	350.00	0.00	350.00

Мильфей с клубникой и кремом "Сабайон"
 1 пор x 350.00 - (0.00 + 0.00) = 350.00

Итого: 1 650.00

Вид меню Рестарт

Блюда из мяса Блюда из рыбы и морепродуктов Гарниры Десерты Свободное блюдо

Оплата Сервис Карта Удалить Скидка Кол-во Пробить чек

Рисунок 1.1 – Вигляд вікна адміністратора в системі АС «Рестарт»

Серед технічних особливостей системи адміністрування АС «Рестарт» можна виділити:

- простий та інтуїтивно зрозумілий інтерфейс, призначений для роботи з сенсорним екраном;
- можливість переключатися між вікнами системи;
- окремі поля для кожної функціональної одиниці;
- наявність кнопок для додавання/редагування/видалення вмісту вікна.

На основі саме цих особливостей й буде спроектовано модель запропонованої системи управління закладами харчування.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Викладення вимог до системи

Проаналізувавши наявні на ринку рішення та дослідивши їх головні особливості, було сформовано основні вимоги до запропонованої системи адміністрування закладом харчування.

Система має відповідати наступним функціональним вимогам:

- у програмі має бути реалізований інтерфейс гостя для пошуку страв по різних категоріях;
- у програмі має бути реалізований інтерфейс адміністратора для редагування меню;
- програма має відображати замовлення кожного окремого клієнта та підраховувати вартість його замовлення;
- програма має в порядку замовлення відображати кількість клієнтів, що замовили ту чи іншу страву для оптимізації роботи кухаря;
- звичайний користувач повинен мати можливість додавати та видаляти користувачів;
- користувач повинен мати можливість додавати та видаляти страву зі свого замовлення;
- адміністратор повинен мати можливість додавати/редагувати/видаляти користувачів та меню.

2.2 Моделювання сутностей, їх атрибутів та зв'язків між ними

Розроблена програма має включати наступні класи: базовий клас, від якого будуть наслідуватись усі інші класи, клас користувача, страви, розв'язочний клас для користувача та страви, клас типу кухні, типу страви та інгредієнта. Рисунок 2.1 ілюструє ієрархію класів та визначені атрибути кожного з них.

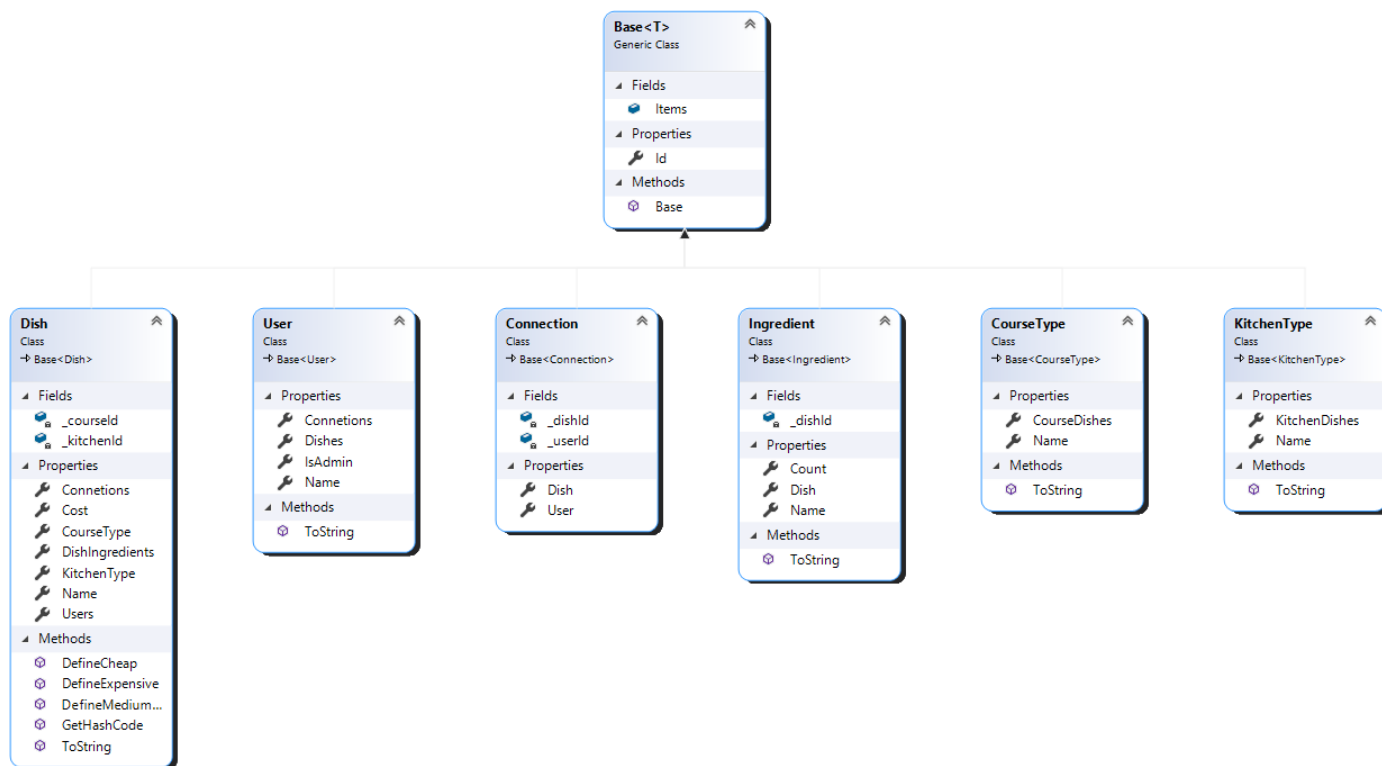


Рисунок 2.1 – Діаграма класів

На вершині цієї ієрархії стоїть батьківський Generic клас Base (дивись таблицю 2.1). Його єдиним атрибутом є Id. Також цей клас має словник Items, що буде використаний у дочірніх класах та конструктор класа. Усі наступні класи наслідують клас Base.

Таблиця 2.1 – Таблиця класу Generic Base

Назва	Тип	Призначення
Id	Guid	Унікальний ідентифікатор (ключ) сутностей
Items	Field	Посилання на об'єкт
Base	ctor	Конструктор класу

Клас Dish (дивись таблицю 2.2) має атрибути: Назва, Тип кухні, Тип страви, список інгредієнтів, список користувачів, що замовили дану страву, Вартість. Клас страви зберігає посилання на класи типу кухні та типу страви. Також він має 3 статичні методи для сортування страв по цінній категорії та перевизначений метод GetHashCode() та ToString().

Таблиця 2.2 – Таблиця класу Dish

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Name	String	Назва страви
KitchenType	KitchenType	Тип кухні
CourseType	CourseType	Тип страви
DishIngredients	List<Ingredient>	Інгредієнти страви
Connections	List<Connections>	Словник посилань на розв'язочний клас
Users	List<User>	Словник користувачів, що замовили страву
Cost	Int	Вартість страви
_kitchenId	Field	Посилання на тип кухні
_courseId	Field	Посилання на тип страви
DefineCheap()	Method	Статичний метод пошуку дешевих страв
DefineMediumCost()	Method	Статичний метод пошуку страв середньої ціни
DefineExpensive()	Method	Статичний метод пошуку дорогих страв
ToString()	Method	Повернення коректного відображення страви
GetHashCode()	Method	Повернення вартості страви

Клас User (дивись таблицю 2.3) має такі атрибути: Ім'я, Чи є користувач адміністратором та список страв, які він замовив. У цьому класі реалізовано перезапис методу ToString().

Таблиця 2.3 – Таблиця класу User

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Name	String	Назва користувача
IsAdmin	Bool	Визначення адміністратора
Dishes	List<Dish>	Словник страв, що замовив користувач
Connections	List<Connections>	Словник посилань на розв'язочний клас
ToString()	Method	Повернення коректного відображення користувача

Клас Connections – це розв'язочний клас (дивись таблицю 2.4), що зберігає в собі посилання на страву та користувача. За допомогою цього класу реалізовано зв'язок багато до багатьох. Цей клас має атрибути: Страва та Користувач.

Таблиця 2.4 – Таблиця класу Connections

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Dish	Dish	Страви, що замовив користувач
User	User	Користувачі, що замовили страву
_dishId	Guid	Посилання на страву
_userId	Guid	Посилання на користувача

Клас Ingredient (дивись таблицю 2.5) включає в себе посилання на страву та має наступні атрибути: Назва, Кількість та страва, до якої він належить. У цьому класі перезаписано метод ToString().

Таблиця 2.5 – Таблиця класу Ingredient

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Name	String	Назва інгредієнту
Count	String	Кількість інгредієнту
Dish	Dish	Страва, до якої належить інгредієнт
_dishId	Guid	Посилання на страву
ToString()	Method	Повернення коректного відображення інгредієнту

Клас CourseType (дивись таблицю 2.6) має атрибути: Назва та список страв, що належать до даного типу страви. Даний клас також реалізує перезапис віртуального методу ToString().

Таблиця 2.6 – Таблиця класу CourseType

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Name	String	Назва типу страви
CourseDishes	List<Dish>	Страви, що належать до типу страви
ToString()	Method	Повернення коректного відображення типу страви

Клас KitchenType (дивись таблицю 2.7) серед атрибутів також має назву та список страв, що належать до даної кухні. У цьому класі також перезаписано метод ToString().

Таблиця 2.7 – Таблиця класу KitchenType

Назва	Тип	Призначення
Id	Guid	Ключ сутності
Name	String	Назва типу кухні
KitchenDishes	List<Dish>	Страви, що належать до типу кухні
ToString()	Method	Повернення коректного відображення типу кухні

Реалізація зв'язків:

- Сутність *Страва*, сутність *Тип кухні*

Сутність *Страва* пов'язана з сутністю *Тип кухні* за допомогою зв'язку багато до одного *:1, тобто кожна *Страва* може мати не більше одного *Типу кухні*, а кожен *Тип кухні* може вміщати багато *Страв*.

- Сутність *Страва*, сутність *Тип страви*

Сутність *Страва* пов'язана з сутністю *Тип страви* за допомогою зв'язку багато до одного *:1, тобто кожна *Страва* може мати не більше одного *Типу страви*, а кожен *Тип страви* може вміщувати багато *Страв*.

- Сутність *Інгредієнт*, сутність *Страва*

Сутність *Інгредієнт* пов'язана з сутністю *Страва* за допомогою зв'язку багато до одного *:1, тобто кожен *Інгредієнт* може бути використаним не більш, як в одній *Страві*, а кожна *Страва* може мати багато *Інгредієнтів*.

- Сутність *Страва*, сутність *Користувач*

Сутність *Страва* пов'язана з сутністю *Користувач* за допомогою зв'язку багато до багатьох *:*, тобто кожна *Страва* може мати безліч *Користувачів*, що її замовили, і кожен *Користувач* може замовити безліч *Страв*.

2.3 Опис сценарії виконання системи

У програмі мають бути наявні наступні сценарії виконання (дивись рисунок 2.2).

Акторами є користувачі системи: користувач без прав адміністратора (Користувач) та користувач з правами адміністратора (Адміністратор). Адміністратору доступен увесь функціонал, що й звичайному користувачу, а також можливість уносити зміни в меню та редагувати інформацію про користувача.



Рисунок 2.2 – Діаграма прецедентів

3 РЕАЛІЗАЦІЯ ОСНОВНИХ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура проекту

Загальна структура проекту зображена на рисунку 3.1.

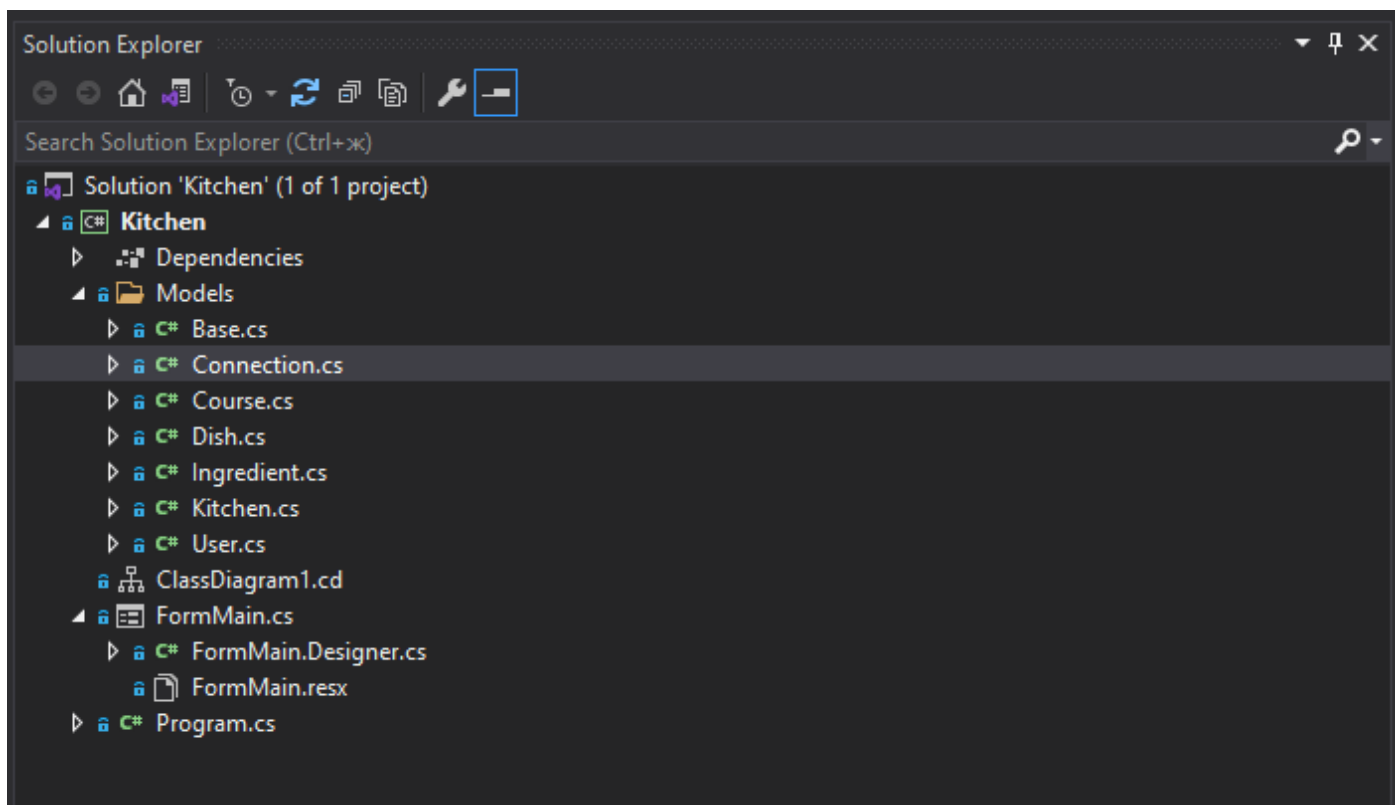


Рисунок 3.1 – Дерево проекту

Проект створено в інтегрованому середовищі розробки програмного забезпечення Microsoft Visual Studio 2019. У проекті для кожної сутності було створено окремий клас, які в свою чергу для зручності було поміщено в окрему папку Models. Проект включає в себе класи сутностей, форму Windows Forms та програмний файл Program.cs.

При створенні проекту було використано сервіс контролю версій Git 4. Готовий проект наявний у публічному репозиторії GitHub 5.

3.2 Реалізація зв'язків багато до одного

Створена програма адміністрування закладами харчування потребує роботи з базами даних. Тому, з метою спрощення реалізації, у програмі було імітовано роботу з таблицями. Тобто, між зазначеними у пункті 2.2 сутностями було реалізовано зв'язки багато до одного та багато до багатьох. Розглянемо детально реалізацію зв'язків багато до одного (далі *:1).

Зв'язок *:1 використовується, якщо одному екземпляру однієї сутності може відповідати будь-яка кількість екземплярів іншої сутності. У рамках нашої задачі такий зв'язок мають: сутності *Страва* - *Тип кухні*, *Страва* - *Тип страви*, *Інгредієнт* - *Страва*. Детальніше ці зв'язки описані у пункті 2.2.

Основні елементи зв'язку *:1:

Розглянемо реалізацію даного зв'язку на прикладі сутностей *Інгредієнт* – *Страва*. У реалізації даного типу зв'язку можна виділити наступні елементи: посилання на іншу сутність, пряма та зворотні властивості.

Реалізація посилання:

Посилання на іншу сутність має міститись у тій сутності, екземплярів якої може бути багато. Тобто, для нашої задачі, в класі *Інгредієнт* знаходиться посилання на клас *Страва* (дивись рисунок 3.2).

```

10 // Ініціалізатор властивості для вмісту інгредієнтів у страві
11 private Guid _dishId;
12
13 44 references | zerorchik, 29 days ago | 1 author, 1 change
14 public Dish Dish
15 {
16     get { return Dish.Items[_dishId]; }
17     set { _dishId = value.Id; }
18 }
```

Рисунок 3.2 – Посилання на ключ сутності *Страва* у класі сутності *Інгредієнт* та реалізація прямої властивості зв'язку *:1

Для правильної реалізації зв'язку *:1 необхідно додати його пряму та зворотню властивості.

Реалізація прямої властивості:

Пряма властивість знаходиться в тому ж класі, що й посилання на іншу сутність, тобто у класі *Інгредієнта* (дивись рисунок 3.2). Вона отримує посилання на сутність, яка має бути одна, точніше на її ключ і повертає список пов'язаних із нею сутностей, яких може бути багато. Тобто, у нашому випадку, пряма властивість отримує ID *Страви* і повертає список *Інгредієнтів*, пов'язаних з цією *Стравою* за їх ключами.

Реалізація зворотної властивості:

Натомість зворотня властивість знаходиться в іншому класі, класі сутності, яка пов'язана з багатьма сутностями першого класу. У нашому випадку зворотня властивість знаходиться в класі *Страва* (дивись рисунок 3.3). Вона має приймати список пов'язаних з нею елементів іншого класу за ключами.

```

12 // Словник інгредієнтів у страві
13 1 reference | 0 changes | 0 authors, 0 changes
14 public List<Ingredient> DishIngredients
15 {
16     get { return Ingredient.Items.Values.Where(ingredient => ingredient.Dish == this).ToList(); }

```

Рисунок 3.3 – Реалізація зворотної властивості зв'язку *:1

У нашому випадку ця властивість перебирає словник інгредієнтів і вибирає з них ті, що пов'язані з поточною стравою. Аналогічна реалізація зв'язків *:1 присутня й у інших вищезазначених класах.

Також варто зазначити, що відбувається із цими сутностями і зв'язками між ними при їх видаленні та редагуванні.

Видалення зв'язаного елементу:

При видаленні екземпляру сутності *Страва*, яка може містити багато *Інгредієнтів*, спочатку видаляються зв'язки між *Стравою* та *Інгредієнтами*, потім усі *Інгредієнти*, що належать *Страві*, і тільки тоді видаляється сама *Страва* (дивись рисунок 3.4).

```

1 reference | 0 changes | 0 authors, 0 changes
269 private void btnRemoveDish_Click(object sender, EventArgs e)
270 {
271     bool contains = false;
272     if (lbDish.SelectedItem != null)
273     {
274         foreach (User user in User.Items.Values)
275             if (user.Dishes.Contains(lbDish.SelectedItem))
276             {
277                 MessageBox.Show("Ви не можете видалити страву, яка вже замовлена!");
278                 contains = true;
279                 break;
280             }
281
282         if (!contains)
283         {
284             int count = lbIngredients.Items.Count;
285             for (int i = 0; i < count; i++)
286                 btnRemoveIngredient_Click(sender, e);
287             Dish.Items.Remove(((Dish)lbDish.SelectedItem).Id);
288             RefreshDishes();
289         }
290     }
291 }

```

```

2 references | 0 changes | 0 authors, 0 changes
258 private void btnRemoveIngredient_Click(object sender, EventArgs e)
259 {
260     var dish = lbDish.SelectedItem;
261     var ingredientToDel = Ingredient.Items.Values.Where(connection => connection.Dish == dish).FirstOrDefault();
262     if (ingredientToDel != null)
263     {
264         Ingredient.Items.Remove(ingredientToDel.Id);
265         RefreshIngredients();
266     }
267 }

```

Рисунок 3.4 – Реалізація видалення сутностей, пов'язаних між собою зв'язком *:1

Редагування зв'язаного елементу:

При редагуванні сутностей змінюються їх атрибути, натомість їх ключі залишаються незмінними. Отже ця операція ніяким чином не впливає на зв'язок між сутностями.

3.3 Реалізація зв'язків багато до багатьох

Зв'язок багато до багатьох (далі *:*) використовується, якщо декільком екземплярам однієї сутності може відповідати декілька екземплярів іншої сутності. У рамках нашої задачі такий зв'язок мають сутності *Страва* та *Користувач*. Детальніше цей зв'язок описаний у пункті 2.2.

Основні елементи зв'язку *:*:

Для реалізації даного типу зв'язку використовують так званий розв'язочний клас 7, який розділяє зв'язок *:* на 2 зв'язки *:1 (дивись рисунок 3.5).

Усі елементи розв'язочного класу являють собою факт зв'язку між двома сутностями. Тобто розв'язочний клас містить у собі унікальні ключі із кожного з двох пов'язаних класів. Таким чином пов'язані цим зв'язком класи будуть мати списки не екземплярів тих сутностей, з якими вони пов'язані, а екземпляри елементів розв'язочного класу. У нашому випадку класи *Страва* та *Користувач* будуть вміщувати словник екземплярів розв'язочного класу.

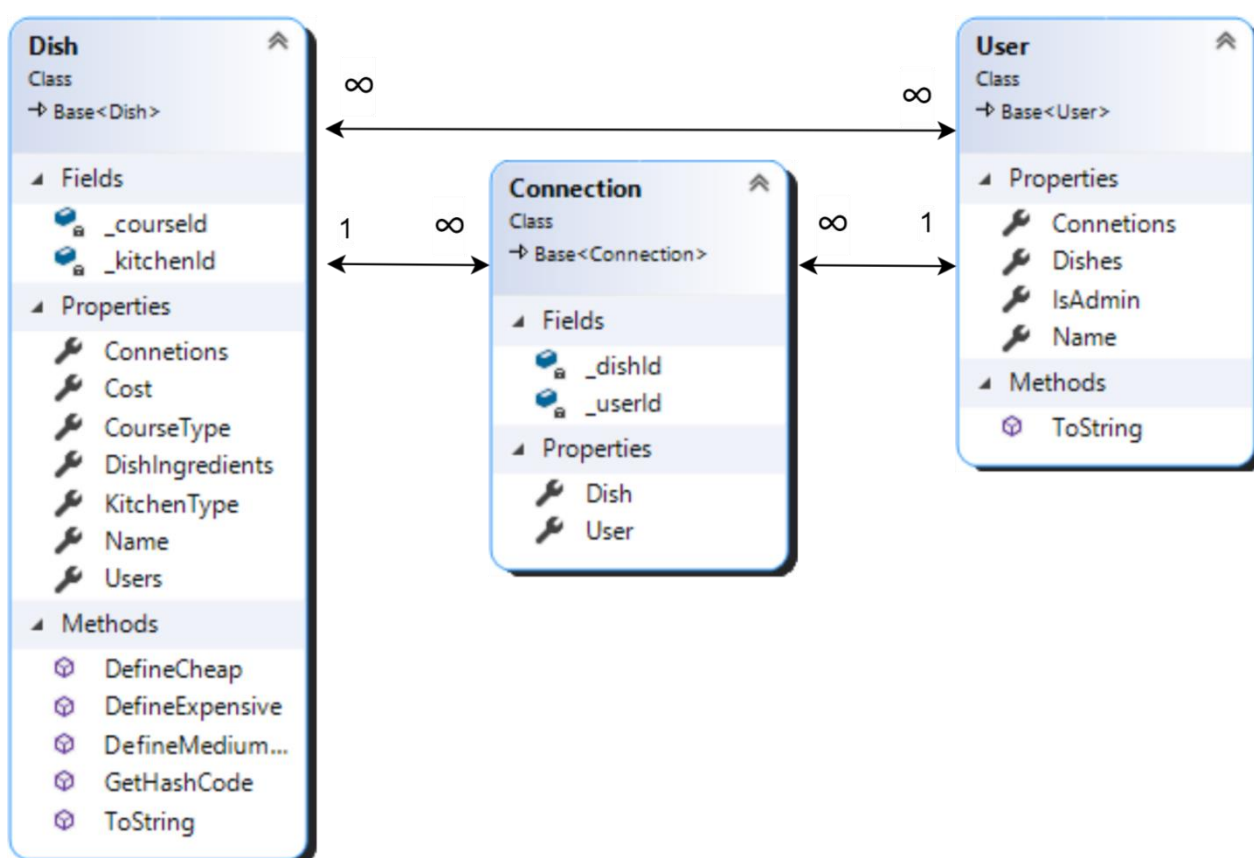


Рисунок 3.5 – Діаграма представлення зв'язку *:*

Реалізація посилання та прямої властивості:

З рисунку 3.5 видно, що розв'язочний клас *Зв'язки* має посилання на класи *Страва* та *Користувач* та реалізовує прямі властивості для цих класів (дивись рисунок 3.6).

Реалізація зворотної властивості:

Натомість класи *Страва* та *Користувач* реалізують зворотню властивість, тобто мають словники елементів класу *Зв'язки* (дивись рисунок 3.7).

Реалізація діставання зі зворотної властивості словника елементів зв'язаного класу:

Кінцевою метою створення такого зв'язку є наявність у класах *Страва* та *Користувач* словників екземплярів пов'язаних класів. Тому у класах *Страва* та *Користувач* реалізовано властивості, що отримують з елементу розв'язочного класу *ID* прив'язаних *Користувачів* та *Страв* відповідно (дивись рисунок 3.7).

```

7      // Ініціалізатор властивості для вмісту страв у користувачі
8      private Guid _userId;
9
10     5 references | 0 changes | 0 authors, 0 changes
11     public User User
12     {
13         get { return User.Items[_userId]; }
14         set { _userId = value.Id; }
15     }
16
17     // Ініціалізатор властивості для вмісту користувачів у страві
18     private Guid _dishId;
19     5 references | 0 changes | 0 authors, 0 changes
20     public Dish Dish
21     {
22         get { return Dish.Items[_dishId]; }
23         set { _dishId = value.Id; }
24     }

```

Рисунок 3.6 – Посилання на ключі сутностей пов'язаних зв'язком **:** та реалізація його прямої властивості через розв'язочний клас

```

18     // Словник зв'язків зі стравою
19     0 references | 0 changes | 0 authors, 0 changes
20     public List<Connection> Connections
21     {
22         get { return Connection.Items.Values.Where(connection => connection.Dish == this).ToList(); }
23     }
24
25     // Словник користувачів у страві
26     1 reference | 0 changes | 0 authors, 0 changes
27     public List<User> Users
28     {
29         get { return Connection.Items.Values.Where(connection => connection.Dish == this).Select(connection => connection.User).ToList(); }
30     }

```

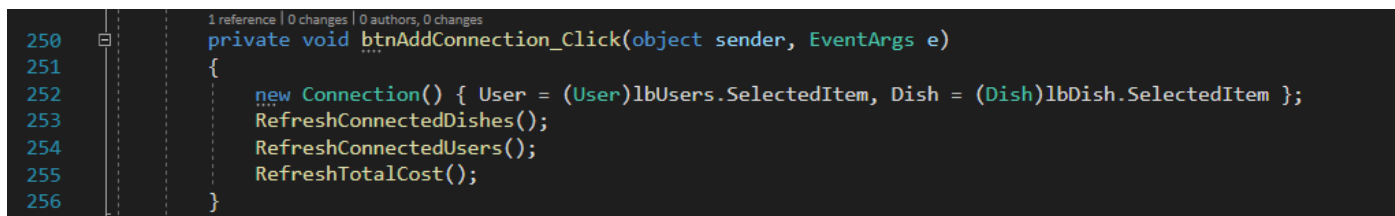
Рисунок 3.7 – Реалізація зворотної властивості зв'язку **:** між *Стравою* та *Користувачем* у класі сутності *Страва* через клас *Зв'язки* та реалізація діставання з розв'язочного класу словника екземплярів сутностей *Користувач* у класі *Страва*

Аналогічна рисунку 3.7 реалізація зворотної властивості зв'язку *:1 присутня й у класі *Користувач*.

Варто також зазначити, що для сутностей пов'язаних зв'язком *:1 операції додавання та видалення екземпляру сутності 8 будуть відрізнятися від аналогічних операцій з сутностями, пов'язаних зв'язком *:1. Натомість, операція редагування екземпляру такої сутності не буде відрізнятися від подібної для зв'язку *:1, оскільки при редагуванні ключ сутності не змінюється. Також, якщо для зв'язку *:1 розглядалось лише видалення пов'язаних між собою елементів, то для зв'язку *:1 є сенс розглянути також окремо створення та видалення зв'язків між ними, так, як вони являють собою окрему сутність.

Створення зв'язку *:1:

Для того, щоб зв'язати між собою 2 сутності: *Користувача* та *Страву* потрібно створити об'єкт розв'язочного класу і вказати для нього обраних *Страву* та *Користувача* (дивись рисунок 3.8).



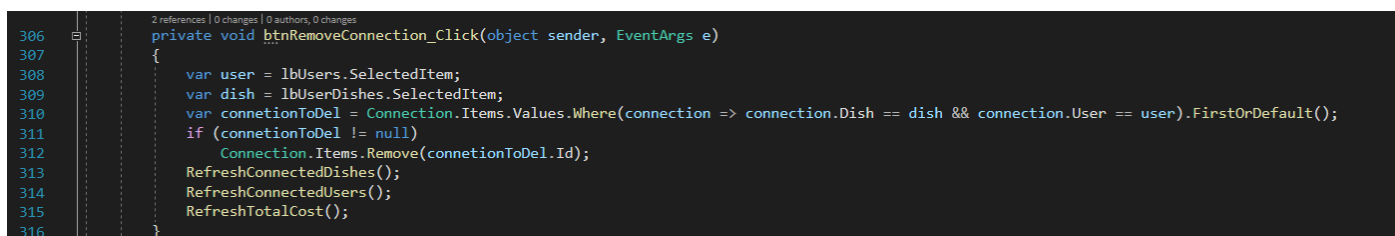
```

250 private void btnAddConnection_Click(object sender, EventArgs e)
251 {
252     new Connection() { User = (User)lbUsers.SelectedItem, Dish = (Dish)lbDish.SelectedItem };
253     RefreshConnectedDishes();
254     RefreshConnectedUsers();
255     RefreshTotalCost();
256 }
  
```

Рисунок 3.8 – Реалізація додавання зв'язку *:1 між сутностями *Страва* та *Користувач*

Видалення зв'язку *:1:

Для того, щоб видалити зв'язок між *Стравою* та *Користувачем* спочатку потрібно віднайти потрібних *Страву* та *Користувача* і видалити відповідний об'єкт класу *Зв'язки*, що їх містить (дивись рисунок 3.9).



```

306 private void btnRemoveConnection_Click(object sender, EventArgs e)
307 {
308     var user = lbUsers.SelectedItem;
309     var dish = lbUserDishes.SelectedItem;
310     var connectionToDel = Connection.Items.Values.Where(connection => connection.Dish == dish && connection.User == user).FirstOrDefault();
311     if (connectionToDel != null)
312         Connection.Items.Remove(connectionToDel.Id);
313     RefreshConnectedDishes();
314     RefreshConnectedUsers();
315     RefreshTotalCost();
316 }
  
```

Рисунок 3.9 – Реалізація видалення зв'язку *:1 між сутностями *Страва* та *Користувач*

Видалення зв'язаного елементу:

Натомість, аби видалити поєднану зв'язком ***:*** Страву або Користувача слід спершу видалити зв'язки між ними (дивись рисунок 3.10). Проте в контексті нашої задачі недоречно видаляти з меню страву, яку вже замовив якийсь користувач, тому в такому випадку програма виведе повідомлення про помилку (дивись рисунок 3.11).

```

293 private void btnRemoveUser_Click(object sender, EventArgs e)
294 {
295     if (lbUsers.SelectedItem != null)
296     {
297         int count = lbUserDishes.Items.Count;
298         for (int i = 0; i < count; i++)
299             btnRemoveConnection_Click(sender, e);
300         User.Items.Remove(((User)lbUsers.SelectedItem).Id);
301         RefreshUsers();
302         RefreshConnectedDishes();
303     }
304 }

```

Рисунок 3.10 – Реалізація видалення пов'язаного зі *Стравою Користувача*

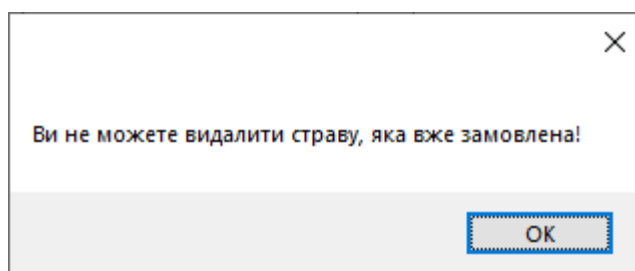


Рисунок 3.11 – Виведення на екран повідомлення про помилку

3.4 Реалізація статичних методів

Ще одним обов'язковим аспектом вивчення ООП є розуміння та реалізація статичних методів. Статичні методи реалізуються в деяких класах і використовуються, коли потрібно використати деякі дані цього класу незалежно від будь-якого його об'єкту. Тобто використання статичних методів не потребує створення об'єкту цього класу.

У рамках поставленої задачі такі методи чудово підійдуть для реалізації пошуку елементів класу Страва по їх цінovій категорії. Для цього було створено 3 статичні методи: пошуку страви у високій цінovій категорії (вартість страви ≥ 100 грн), середній ($50 \leq$ вартість страви < 100 грн) та дешевій (вартість страви < 50 грн) цінovих категоріях (дивись рисунок 3.12).

```

48 // Реалізація трьох статичних методів
49 1 reference | 0 changes | 0 authors, 0 changes
50 public static bool DefineCheap(Dish dishes)
51 {
52     if (dishes.Cost < 50) return true;
53     return false;
54 }
55 1 reference | 0 changes | 0 authors, 0 changes
56 public static bool DefineMediumCost(Dish dishes)
57 {
58     if (dishes.Cost >= 50 && dishes.Cost < 100) return true;
59     return false;
60 }
61 1 reference | 0 changes | 0 authors, 0 changes
62 public static bool DefineExpensive(Dish dishes)
63 {
64     if (dishes.Cost >= 100) return true;
65     return false;
66 }

```

Рисунок 3.12 – Реалізація статичних методів пошуку *Страви* по цінових категоріях

Дані методи реалізовано так, що вони перебирають список усіх страв і визначають, чи задовольняє кожна страва заданим критеріям пошуку. Якщо умова пошуку виконується, то страва додається до списку страв вказаної категорії (дивись рисунок 3.13).

```

325 1 reference | 0 changes | 0 authors, 0 changes
326 private void btnCheap_Click(object sender, EventArgs e)
327 {
328     ClearLbDish();
329     foreach (Dish dish in Dish.Items.Values)
330         if (Dish.DefineCheap(dish))
331             lbDish.Items.Add(dish);
332 }
333 1 reference | 0 changes | 0 authors, 0 changes
334 private void btnMiddle_Click(object sender, EventArgs e)
335 {
336     ClearLbDish();
337     foreach (Dish dish in Dish.Items.Values)
338         if (Dish.DefineMediumCost(dish))
339             lbDish.Items.Add(dish);
340 }
341 1 reference | 0 changes | 0 authors, 0 changes
342 private void btnExpensive_Click(object sender, EventArgs e)
343 {
344     ClearLbDish();
345     foreach (Dish dish in Dish.Items.Values)
346         if (Dish.DefineExpensive(dish))
347             lbDish.Items.Add(dish);
348 }

```

Рисунок 3.13 – Реалізація виведення результату про виконання статичних методів

3.5 Користувачський інтерфейс та діалоги користувача

Для правильної та зручної роботи розробленої системи адміністрування закладами харчування потрібно реалізувати зрозумілий інтерфейс користувача. Вікно реалізованого користувачького інтерфейсу зображає рисунок 3.14.

Відповідно до сформованих при аналізі конкурентних систем вимог до користувачького інтерфейсу, було прийнято рішення зону для відображення зв'язків між сутностями, зону редагування сутностей та зв'язків між ними та зону пошуку екземплярів сутності *Страва* по її характеристикам.

Відображення зв'язків: Пошук по:

Клієнти:	Обрані страви:	Прив'язані клієнти:	Кухні народів світу:	Страви:	Інгредієнти:
Сашко (адмін) Микола	Борщ - 85 Мотті - 35	Сашко (адмін) Микола	Українська кухня	Борщ - 85 Рамен - 120 Вареники з картоплею - 55 Цезар - 45 Кола - 20 Мотті - 35	* буряк - 2 штуки * морква - 1 штука * цибуля - 3 штуки * томатна паста - 2 ст. ложки * соняшникова олія - 4-5 ст. ложок * лимонна кислота - дрібка

Загалом: 120 грн

Додати | Редагувати | Видалити | Замовити | Вилучити з замовлення

Уведіть клієнта: Сашко
☒ адмін

Оберіть кухню: Українська кухня
Оберіть тип блюда: Перша страва

Уведіть назву страви: Борщ
Уведіть її вартість: 85

Уведіть інгредієнт: буряк
Уведіть його кількість: 2 штуки

Додати | Редагувати | Видалити

Рисунок 3.14 – Вікно інтерфейсу користувача

Обґрунтування створення зазначеного інтерфейсу:

Для подальшого вдосконалення системи, при її масштабуванні, для зручності користування, можна виділити під кожну сутність окрему форму. Проте, на даній стадії проекту, при роботі з малим обсягом даних зручніше буде працювати в одному вікні. Також, таке рішення є простішим у реалізації, тому врешті зупинимось на ньому.

Варто ще зазначити, що для реальної системи обліку рестораном необхідно реалізувати окремі форми для клієнта закладу та його персоналу. Утім, для запропонованої моделі такої системи буде достатньо продемонструвати працездатність програми і реалізувати їх функціонал у одному вікні.

Проте, оскільки реалізацію користувацького інтерфейсу було строщено, варто розбити головне функціональне вікно на вищезазначені підзони за способом їх використання.

Зона відображення зв'язків:

Перша зона буде відповідати за відображення зв'язків (дивись рисунок 3.15).

Recepies

Відображення зв'язків:

Пошук по:

Клієнти:

- Сашко (адмін)
- Микола

Обрані страви:

Тут будуть відображатися страви, зв'язані з обраним користувачем

Прив'язані клієнти:

Тут будуть відображатися користувачі, зв'язані з обраною стравою

Кухні народів світу:

Українська кухня

Тип блюда:

Перша страва

Усі страви

Дешеві

Середньої ціни

Дорогі

Страви:

- Борщ - 85
- Рамен - 120
- Вареники з картоплею - 55
- Цезар - 45
- Кола - 20
- Мотті - 35

Інгредієнти:

- * буряк - 2 штуки
- * морква - 1 штука
- * цибуля - 3 штуки
- * томатна паста - 2 ст. ложки
- * соняшникова олія - 4-5 ст. ложок
- * лимонна кислота - дрібка

Уведіть клієнта:

Сашко

☒ адмін

Додати

Редагувати

Видалити

Загалом:

0 грн

Замовити

Вилучити з замовлення

Оберіть кухню:

Українська кухня

Оберіть тип блюда:

Перша страва

Додати

Редагувати

Видалити

Уведіть назву страви:

Борщ

Уведіть її вартість:

85

Додати

Редагувати

Видалити

Уведіть інгредієнт:

буряк

Уведіть його кількість:

2 штуки

Додати

Редагувати

Видалити

Рисунок 3.15 – Зображення зони відображення зв'язків між сутностями

З рисунку 3.15 видно, що зазначена зона відображає зв'язки *:1 між сутностями *Страва* та *Користувач* та зв'язки *:1 між сутностями *Страва* – *Тип кухні*, *Страва* – *Тип страви*, *Інгредієнт* – *Страва*.

Елементи позначені сірими прямокутниками не відносяться до даної зони. Стрілочки показують зв'язок яких списків відображає даний список. Бачимо, що на демонстрацію зв'язків *:1 виділено 2 списки, а для демонстрації зв'язків *:1 у нижній частині екрану наявні 2 випадаючі списки, що демонструють до якого *Типу кухні* та *Типу страви* відноситься дана *Страва*, та список *Інгредієнтів*, що їй належать. Також у цій зоні наявні 2 кнопки для створення та видалення зв'язків *:1, описані раніше.

Зона пошуку Страв:

Наступною розглянемо область пошуку *Страв* за їх атрибутами, зображену на рисунку 3.16.

Пошук по:

Кухні народів світу: Українська кухня ▼

Тип блюда: Перша страва ▼

Страви:

- Борщ - 85
- Рамен - 120
- Вареники з картоплею - 55
- Цезар - 45
- Кола - 20
- Мотті - 35

Усі страви

Дешеві

Середньої ціни

Дорогі

Рисунок 3.16 – Зображення області пошуку *Страв* за атрибутами

З рисунку 3.16 видно, що пошук можна здійснити по *Типу кухні*, *Типу страви* та за допомогою раніше описаних статичних методів, також можна переглянути все меню.

Можна помітити, що при пошуку страв за атрибутами у списку *Страв* відображаються зв'язки *:1 між *Стравою*, її *Типом кухні* та *Типом страви* у зворотному порядку від того, як вони відображаються у зоні відображення зв'язків. Тобто, задіяна властивість отримання *Типу кухні* та *Типу страви* для обраної *Страви* за її ID.

Зона редагування:

Наступною розглянемо область екземплярів усіх наявних сутностей, зображену на рисунку 3.17.

Рисунок 3.17 – Зображення області редагування екземплярів вказаних сутностей

З рисунку 3.17 видно, що дана зона здебільшого містить кнопки для додавання/редагування/видалення сутностей. Також у цій зоні наявні поля для вводу тексту. У них для зручності відображаються атрибути обраних у списках екземплярів сутностей. Ці поля використовуються при додаванні нових та редагуванні вже наявних екземплярів сутностей. Для відображення атрибуту Користувача «є адміном» у заданій області використовується checkbox, який має 2 стани: увімкнено та вимкнено. Сірі прямокутники позначають елементи, що не входять до заданої зони.

Вигляд інтерфейсу користувача не для адміністратора:

Варто вказати, що інтерфейс користувача, зображений на рисунку 3.14 є актуальним для *Користувача* з правами адміністратора. Оскільки адміністратор у даній системі може як редагувати меню, так і робити замовлення. Відповідно, якщо обраний у списку *Користувачів* *Користувач* не є адміністратором, то йому недоступна область редагування (дивись рисунок 3.18).

Відображення зв'язків: Пошук по:

Клієнти:

- Сашко (адмін)
- Микола**

Обрані страви:

Прив'язані клієнти:

Кухні народів світу:

Українська кухня

Тип блюда:

Перша страва

Усі страви

Дешеві

Середньої ціни

Дорогі

Страви:

- Борщ - 85**
- Рамен - 120
- Вареники з картоплею - 55
- Цезар - 45
- Кола - 20
- Мотті - 35

Інгредієнти:

- * бурак - 2 штуки**
- * морква - 1 штука
- * цибуля - 3 штуки
- * томатна паста - 2 ст. ложки
- * соняшникова олія - 4-5 ст. ложок
- * лимонна кислота - дрібка

Уведіть клієнта:

Микола

☐ адмін

Загалом:

0 грн

Оберіть кухню:

Українська кухня

Оберіть тип блюда:

Перша страва

Уведіть назву страви:

Борщ

Уведіть її вартість:

85

Уведіть інгредієнт:

бурак

Уведіть його кількість:

2 штуки

Додати **Замовити**

Редагувати

Видалити

Видалити з замовлення

Додати **Додати**

Редагувати

Видалити

Видалити

Рисунок 3.18 – Вигляд інтерфейсу користувача для не адміністратора

Утім, можна помітити, що *Користувач* без прав адміністратора може додавати та видаляти користувачів, оскільки, якщо цю функцію заборонити, рано чи пізно це призведе до того, що всі адміністратори будуть видалені, а додати нових *Користувачів* звичайний *Користувач* не може.

3.6 Перспективи покращення системи

Як було вказано в пункті 3.5, реалізованої системи достатньо на даному етапі побудови проекту, оскільки її головна задача – це демонстрація працездатності її основних функціональних компонентів.

Утім, розробленій системі є куди рости. У разі переходу цього проекту від стадії проектування до стадії реалізації мінімально життєздатного продукту (MVP) потрібно буде реалізувати зв'язки між сутностями за допомогою баз даних. Попередню підготовку до цього етапу було проведено: усі сутності було розміщено в окремих класах, з метою швидкого розділення їх згодом на окремі таблиці. Також потрібно буде винести функціонал звичайного *Користувача* (не адміністратора) та адміністратора у дві окремі форми. Також, у подальшому бажано було б виділити окремий клас для адміністратора, так як у подальшому в нього може з'явитись більше функціоналу відмінного від звичайного *Користувача*.

ВИСНОВКИ

Був проведений ретельний аналіз існуючих систем автоматизованого управління закладами харчування, у результаті якого було виявлено головні їх технічні особливості. На підставі цих особливостей та з метою отримання практичного досвіду в створенні повноцінних проектів на мові програмування C# і засвоєння основних принципів ООП, був розроблений концепт автономної системи управління закладом харчування. Розроблена система включає в себе інтерфейс користувача для пошуку страв за різними критеріями та інтерфейс адміністратора, що може вносити зміни в меню.

Створена робоча модель запропонованої системи у середовищі Windows Forms. У програмі реалізовано наслідування та поліморфізм. Між класами Користувач та Страва реалізовано зв'язок багато до багатьох, а між класами Страва та Тип кухні, Страва та Тип страви, Інгредієнт та Страва реалізовано зв'язки багато до одного. У класі Страва реалізовано 3 статичні методи для пошуку страв за їх ціновою категорією.

Розроблену систему було протестовано на наявність збоїв у роботі програми та усунено видимі джерела помилок. Також у розробленій системі передбачена можливість подальшого її розширення та покращення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архіпов В.В. Організація ресторанного господарства: навчальний посібник. Київ: Центр учбової літератури, Фірма «Інкос» 2007. 335 с.
2. Пророчук Ж.А., Журавлєва А. Роль програмного забезпечення в управлінні підприємством. *Современные информационные технологии*. 2010. Вип. 1. С.18-20.
3. Програмний продукт «РестАрт». *Medias*.
URL: https://m.medias.com.ua/catalog/programni_produkty_1C/RESTART/
(дата звернення 05.05.2022)
4. GitHub Головна. *GitHub*. URL: <https://github.com>
5. Kursach-proga-2 zerorchik's public repository. *GitHub*.
URL: <https://github.com/zerorchik/Kursach-proga-2>
6. Хмелюк В.С. Зв'язок багато до одного. *YouTube*.
URL: <https://youtu.be/nB34TZTMK3Y> (дата звернення 05.05.2022)
7. Хмелюк В.С. Зв'язок багато до багатьох ч1. *YouTube*.
URL: <https://youtu.be/opBHQaOkF1Y> (дата звернення 05.05.2022)
8. Хмелюк В.С. Зв'язок багато до багатьох ч2. *YouTube*.
URL: https://youtu.be/Qmo7-C1f_kQ (дата звернення 05.05.2022)

ДОДАТОК А

Текст програми

```

0 references | 0 changes | 0 authors, 0 changes
public class Base<T> where T : Base<T>
{
    public static Dictionary<Guid, T> Items = new Dictionary<Guid, T>();

    11 references | 0 changes | 0 authors, 0 changes
    public Guid Id { get; private set; }

    0 references | 0 changes | 0 authors, 0 changes
    public Base()
    {
        Id = Guid.NewGuid();
        Items.Add(Id, (T)this);
    }
}

```

Рисунок А.1 – Текст класу Base

```

19 references | 1 changes | 1 authors, 1 changes
public class User : Base<User>
{
    6 references | 1 changes | 1 authors, 1 changes
    public string Name { get; set; }
    7 references | 1 changes | 1 authors, 1 changes
    public bool IsAdmin { get; set; }

    // Словник зв'язків з користувачем
    0 references | 1 changes | 1 authors, 1 changes
    public List<Connection> Connections
    {
        get { return Connection.Items.Values.Where(connection => connection.User == this).ToList(); }
    }

    // Словник страв у користувачі
    2 references | 1 changes | 1 authors, 1 changes
    public List<Dish> Dishes
    {
        get { return Connection.Items.Values.Where(connection => connection.User == this).Select(connection => connection.Dish).ToList(); }
    }

    0 references | 1 changes | 1 authors, 1 changes
    public override string ToString()
    {
        return Name + (IsAdmin? " (адмін)":"" );
    }
}

```

Рисунок А.2 – Текст класу User

```

44 references | zerorchik, 30 days ago | 1 author, 1 change
public class Dish : Base<Dish>
{
    10 references | zerorchik, 30 days ago | 1 author, 1 change
    public string Name { get; set; }
    15 references | 0 changes | 0 authors, 0 changes
    public int Cost { get; set; }

    // Словник інгредієнтів у страві
    1 reference | 0 changes | 0 authors, 0 changes
    public List<Ingredient> DishIngredients
    {
        get { return Ingredient.Items.Values.Where(ingredient => ingredient.Dish == this).ToList(); }
    }

    // Словник зв'язків зі стравою
    0 references | 0 changes | 0 authors, 0 changes
    public List<Connection> Connections
    {
        get { return Connection.Items.Values.Where(connection => connection.Dish == this).ToList(); }
    }

    // Словник користувачів у страві
    1 reference | 0 changes | 0 authors, 0 changes
    public List<User> Users
    {
        get { return Connection.Items.Values.Where(connection => connection.Dish == this).Select(connection => connection.User).ToList(); }
    }

    // Ініціалізатор властивості для вмісту страв у кухні
    private Guid _kitchenId;

    10 references | 0 changes | 0 authors, 0 changes
    public KitchenType KitchenType
    {
        get { return KitchenType.Items[_kitchenId]; }
        set { _kitchenId = value.Id; }
    }

    // Ініціалізатор властивості для вмісту страв у прийомах їжі
    private Guid _courseId;

    10 references | 0 changes | 0 authors, 0 changes
    public CourseType CourseType
    {
        get { return CourseType.Items[_courseId]; }
        set { _courseId = value.Id; }
    }

    // Реалізація трьох статичних методів
    1 reference | 0 changes | 0 authors, 0 changes
    public static bool DefineCheap(Dish dishes)
    {
        if (dishes.Cost < 50) return true;
        return false;
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public static bool DefineMediumCost(Dish dishes)
    {
        if (dishes.Cost >= 50 && dishes.Cost < 100) return true;
        return false;
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public static bool DefineExpensive(Dish dishes)
    {
        if (dishes.Cost >= 100) return true;
        return false;
    }

    0 references | zerorchik, 30 days ago | 1 author, 1 change
    public override string ToString()
    {
        return Name + " - " + Cost;
    }

    0 references | 0 changes | 0 authors, 0 changes
    public override int GetHashCode()
    {
        return Cost;
    }
}

```

Рисунок А.3 – Текст класу Dish

```

10 references | 0 changes | 0 authors, 0 changes
public class Connection : Base<Connection>
{
    // Ініціалізатор властивості для вмісту страв у користувачі
    private Guid _userId;

    5 references | 0 changes | 0 authors, 0 changes
    public User User
    {
        get { return User.Items[_userId]; }
        set { _userId = value.Id; }
    }

    // Ініціалізатор властивості для вмісту користувачів у страві
    private Guid _dishId;
    5 references | 0 changes | 0 authors, 0 changes
    public Dish Dish
    {
        get { return Dish.Items[_dishId]; }
        set { _dishId = value.Id; }
    }
}

```

Рисунок А.4 – Текст класу Connections

```

14 references | 0 changes | 0 authors, 0 changes
public class KitchenType : Base<KitchenType>
{
    6 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    // Словник страв у кухні
    1 reference | 0 changes | 0 authors, 0 changes
    public List<Dish> KitchenDishes
    {
        get { return Dish.Items.Values.Where(dish => dish.KitchenType == this).ToList(); }
    }

    0 references | 0 changes | 0 authors, 0 changes
    public override string ToString()
    {
        return Name;
    }
}

```

Рисунок А.5 – Текст класу KitchenType

```

15 references | 0 changes | 0 authors, 0 changes
public class CourseType : Base<CourseType>
{
    7 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    // Словник страв у прийомах їжі
    1 reference | 0 changes | 0 authors, 0 changes
    public List<Dish> CourseDishes
    {
        get { return Dish.Items.Values.Where(li => li.CourseType == this).ToList(); }
    }

    0 references | 0 changes | 0 authors, 0 changes
    public override string ToString()
    {
        return Name;
    }
}

```

Рисунок А.6 – Текст класу DishType

```

51 references | zerorchik, 30 days ago | 1 author, 1 change
public class Ingredient : Base<Ingredient>
{
    45 references | zerorchik, 30 days ago | 1 author, 1 change
    public string Name { get; set; }
    45 references | zerorchik, 30 days ago | 1 author, 1 change
    public string Count { get; set; }

    // Ініціалізатор властивості для вмісту інгредієнтів у страві
    private Guid _dishId;

    44 references | zerorchik, 30 days ago | 1 author, 1 change
    public Dish Dish
    {
        get { return Dish.Items[_dishId]; }
        set { _dishId = value.Id; }
    }

    0 references | zerorchik, 30 days ago | 1 author, 1 change
    public override string ToString()
    {
        return "*" + Name + " - " + Count;
    }
}

```

Рисунок А.7 – Текст класу Ingredient

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnAllDishes_Click(object sender, EventArgs e)
{
    RefreshDishes();
}

1 reference | 0 changes | 0 authors, 0 changes
private void lbUsers_SelectedIndexChanged(object sender, EventArgs e)
{
    if (lbUsers.SelectedItem != null)
    {
        // Записуємо у параметри користувача
        tbUserName.Text = ((User)lbUsers.SelectedItem).Name;
        checkBoxIsAdmin.Checked = ((User)lbUsers.SelectedItem).IsAdmin ? true : false;
        // Високоображаємо страву користувача
        RefreshConnectedDishes();
        RefreshTotalCost();
        AdminChecking(((User)lbUsers.SelectedItem).IsAdmin);
    }
}

1 reference | 0 changes | 0 authors, 0 changes
private void AdminChecking(bool decigion)
{
    btnEditUser.Enabled = decigion;
    cbDishKitchen.Enabled = cbDishCourse.Enabled = tbDishName.Enabled = tbDishCost.Enabled = btnAddDish.Enabled = btnEditDish.Enabled = btnRemoveDish.Enabled = decigion;
    tbIngredientName.Enabled = tbIngredientCount.Enabled = btnAddIngredient.Enabled = btnEditIngredient.Enabled = btnRemoveIngredient.Enabled = decigion;
}

```

Рисунок А.8 – Текст функцій додавання страв, зміни вибраного користувача та перевірки на наявність прав адміністратора

```

1 reference | 0 changes | 0 authors, 0 changes
private void lbDish_SelectedIndexChanged(object sender, EventArgs e)
{
    if (lbDish.SelectedItem != null)
    {
        // Записуємо у текстові параметри страви
        tbDishName.Text = ((Dish)lbDish.SelectedItem).Name;
        cbDishKitchen.Text = ((Dish)lbDish.SelectedItem).KitchenType.Name;
        cbDishCourse.Text = ((Dish)lbDish.SelectedItem).CourseType.Name;
        tbDishCost.Text = ((Dish)lbDish.SelectedItem).Cost.ToString();
        // Високоображаємо інгредієнти страви
        RefreshIngredients();
        // Високоображаємо клієнтів, що замовили страву
        RefreshConnectedUsers();
    }
}

1 reference | 0 changes | 0 authors, 0 changes
private void lbIngredients_SelectedIndexChanged(object sender, EventArgs e)
{
    if (lbIngredients.SelectedItem != null)
    {
        tbIngredientName.Text = ((Ingredient)lbIngredients.SelectedItem).Name;
        tbIngredientCount.Text = ((Ingredient)lbIngredients.SelectedItem).Count;
    }
}

1 reference | 0 changes | 0 authors, 0 changes
private void cbKitchen_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbKitchen.SelectedItem != null)
    {
        lbDish.DataSource = null;
        lbDish.DataSource = ((KitchenType)cbKitchen.SelectedItem).KitchenDishes;
    }
}

```

Рисунок А.9 – Текст функцій зміни вибраної страви, інгредієнта та типу кухні

```

1 reference | 0 changes | 0 authors, 0 changes
private void cbCourse_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbCourse.SelectedItem != null)
    {
        lbDish.DataSource = null;
        lbDish.DataSource = ((CourseType)cbCourse.SelectedItem).CourseDishes;
    }
}

4 references | 0 changes | 0 authors, 0 changes
private void RefreshUsers()
{
    lbUsers.DataSource = null;
    lbUsers.DataSource = User.Items.Values.ToList();
}

5 references | 0 changes | 0 authors, 0 changes
private void RefreshDishes()
{
    lbDish.DataSource = null;
    lbDish.DataSource = Dish.Items.Values.ToList();
}

4 references | 0 changes | 0 authors, 0 changes
private void RefreshIngredients()
{
    lbIngredients.DataSource = null;
    lbIngredients.DataSource = ((Dish)lbDish.SelectedItem).DishIngredients;
}

```

Рисунок А.10 – Текст функцій зміни вибраного типу страви, оновлення списку страв, користувачів, інгредієнтів у страві

```

4 references | 0 changes | 0 authors, 0 changes
private void RefreshConnectedDishes()
{
    if (lbUsers.SelectedItem != null)
    {
        lbUserDishes.DataSource = null;
        lbUserDishes.DataSource = ((User)lbUsers.SelectedItem).Dishes;
    }
}

3 references | 0 changes | 0 authors, 0 changes
private void RefreshConnectedUsers()
{
    lbConnectedUser.DataSource = null;
    lbConnectedUser.DataSource = ((Dish)lbDish.SelectedItem).Users;
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnEditDish_Click(object sender, EventArgs e)
{
    KitchenType kitchen;
    CourseType course;
    VarsForDish(out kitchen, out course);
    ((Dish)lbDish.SelectedItem).Name = tbDishName.Text;
    ((Dish)lbDish.SelectedItem).Cost = Convert.ToInt32(tbDishCost.Text);
    ((Dish)lbDish.SelectedItem).KitchenType = kitchen;
    ((Dish)lbDish.SelectedItem).CourseType = course;
    RefreshDishes();
}

```

Рисунок А.11 – Текст функцій оновлення списку прив'язаних до страви користувачів, прив'язаних до користувача страв, редагування атрибутів страви

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnEditIngredient_Click(object sender, EventArgs e)
{
    ((Ingredient)lbIngredients.SelectedItem).Name = tbIngredientName.Text;
    ((Ingredient)lbIngredients.SelectedItem).Count = tbIngredientCount.Text;
    RefreshIngredients();
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnEditUser_Click(object sender, EventArgs e)
{
    ((User)lbUsers.SelectedItem).Name = tbUserName.Text;
    ((User)lbUsers.SelectedItem).IsAdmin = checkBoxIsAdmin.Checked;
    RefreshUsers();
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddDish_Click(object sender, EventArgs e)
{
    KitchenType kitchen;
    CourseType course;
    VarsForDish(out kitchen, out course);
    new Dish() { Name = tbDishName.Text, KitchenType = kitchen, CourseType = course, Cost = Convert.ToInt32(tbDishCost.Text) };
    RefreshDishes();
}

2 references | 0 changes | 0 authors, 0 changes
private void VarsForDish(out KitchenType kitchen, out CourseType course)
{
    kitchen = (KitchenType)cbDishKitchen.SelectedItem;
    course = (CourseType)cbDishCourse.SelectedItem;
}

```

Рисунок А.12 – Текст функцій редагування атрибутів інгредієнта, користувача, додавання користувача

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddIngredient_Click(object sender, EventArgs e)
{
    var dish = (Dish)lbDish.SelectedItem;
    new Ingredient() { Name = tbIngredientName.Text, Count = tbIngredientCount.Text, Dish = dish };
    RefreshIngredients();
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddUser_Click(object sender, EventArgs e)
{
    new User() { Name = tbUserName.Text, IsAdmin = checkBoxIsAdmin.Checked };
    RefreshUsers();
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddConnection_Click(object sender, EventArgs e)
{
    new Connection() { User = (User)lbUsers.SelectedItem, Dish = (Dish)lbDish.SelectedItem };
    RefreshConnectedDishes();
    RefreshConnectedUsers();
    RefreshTotalCost();
}

2 references | 0 changes | 0 authors, 0 changes
private void btnRemoveIngredient_Click(object sender, EventArgs e)
{
    var dish = lbDish.SelectedItem;
    var ingredientToDel = Ingredient.Items.Values.Where(connection => connection.Dish == dish).FirstOrDefault();
    if (ingredientToDel != null)
    {
        Ingredient.Items.Remove(ingredientToDel.Id);
        RefreshIngredients();
    }
}

```

Рисунок А.13 – Текст функцій додавання інгредієнту, користувача, зв'язку між стравою та користувачем, видалення інгредієнту

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnRemoveDish_Click(object sender, EventArgs e)
{
    bool contains = false;
    if (lbDish.SelectedItem != null)
    {
        foreach (User user in User.Items.Values)
        {
            if (user.Dishes.Contains(lbDish.SelectedItem))
            {
                MessageBox.Show("Ви не можете видалити страву, яка вже замовлена!");
                contains = true;
                break;
            }
        }

        if (!contains)
        {
            int count = lbIngredients.Items.Count;
            for (int i = 0; i < count; i++)
            {
                btnRemoveIngredient_Click(sender, e);
            }
            Dish.Items.Remove(((Dish)lbDish.SelectedItem).Id);
            RefreshDishes();
        }
    }
}

```

Рисунок А.14 – Текст функції видалення страви

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnRemoveUser_Click(object sender, EventArgs e)
{
    if (lbUsers.SelectedItem != null)
    {
        int count = lbUserDishes.Items.Count;
        for (int i = 0; i < count; i++)
        {
            btnRemoveConnection_Click(sender, e);
        }
        User.Items.Remove(((User)lbUsers.SelectedItem).Id);
        RefreshUsers();
        RefreshConnectedDishes();
    }
}

2 references | 0 changes | 0 authors, 0 changes
private void btnRemoveConnection_Click(object sender, EventArgs e)
{
    var user = lbUsers.SelectedItem;
    var dish = lbUserDishes.SelectedItem;
    var connectionToDel = Connection.Items.Values.Where(connection => connection.Dish == dish && connection.User == user).FirstOrDefault();
    if (connectionToDel != null)
    {
        Connection.Items.Remove(connectionToDel.Id);
        RefreshConnectedDishes();
        RefreshConnectedUsers();
        RefreshTotalCost();
    }
}

3 references | 0 changes | 0 authors, 0 changes
private void RefreshTotalCost()
{
    int cost = 0;
    foreach (var dishes in lbUserDishes.Items) cost += dishes.GetHashCode();
    labelTotalCost.Text = Convert.ToString(cost) + " грн";
}

```

Рисунок А.15 – Текст функцій видалення користувача, зв'язку між стравою та користувачем, визначення вартості замовлення

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnCheap_Click(object sender, EventArgs e)
{
    ClearLbDish();
    foreach (Dish dish in Dish.Items.Values)
        if (Dish.DefineCheap(dish))
            lbDish.Items.Add(dish);
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnMiddle_Click(object sender, EventArgs e)
{
    ClearLbDish();
    foreach (Dish dish in Dish.Items.Values)
        if (Dish.DefineMediumCost(dish))
            lbDish.Items.Add(dish);
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnExpensive_Click(object sender, EventArgs e)
{
    ClearLbDish();
    foreach (Dish dish in Dish.Items.Values)
        if (Dish.DefineExpensive(dish))
            lbDish.Items.Add(dish);
}

```

Рисунок А.16 – Текст функцій пошуку страв за ціною

```

3 references | 0 changes | 0 authors, 0 changes
private void ClearLbDish()
{
    if(lbDish.DataSource != null)
        lbDish.DataSource = null;
    else
    {
        int count = lbDish.Items.Count;
        for (int ii, i = 0; i < count; i++)
        {
            if (count - i == 1) ii = 0;
            else if (i % 2 == 0) ii = i / 2 + 1;
            else ii = i / 2;
            lbDish.Items.RemoveAt(ii);
        }
    }
}

```

Рисунок А.17 – Текст функції очистки списку страв