МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Програмний додаток підтримки діяльності волонтерської організації

Здобувача (ки) групи _ІТ-91_ _____Дєдовського Дмитра Ігоровича_____
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ __Дмитро ДЄДОВСЬКИЙ_____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник _доц., к.т.н., доц. Світлана ВАЩЕНКО_____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ
_____ Світлана ВАЩЕНКО
«__» _____ 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Дєдовському Дмитру Ігоровичу

1 Тема роботи *Програмний додаток підтримки діяльності волонтерської організації*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « 29 » травня 2023 р. №0588-VI

2 Строк подання студентом роботи «09» червня 2023 р.

3 Вхідні дані до роботи технічне завдання на розробку програмного додатку підтримки діяльності волонтерської організації

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування програмного додатку, розробка програмного додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) вікно авторизації, вікно пошуку замовника, вікно карти замовника, вікно оформлення видачі, вікно додавання нового замовника, вікно додавання нового пакунку, вікно панелі адміністратора, вікно статистики, висновки

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання___10.10.2022____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Дослідження предметної області	20.10.2022- 30.10.2022	
	Розробка макету програмного додатку	10.11.2022- 30.11.2022	
	Розробка програмного додатку	01.12.2022- 24.02.2023	
	Тестування	25.02.2023- 14.03.2023	
	Перевірка працездатності	17.03.2023- 23.03.2023	
	Оформлення пояснювальної записки	24.03.2023- 31.05.2023	

Студент _____ Дмитро ДЄДОВСЬКИЙ
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Програмний додаток підтримки діяльності волонтерської організації».

Пояснювальна записка складається зі вступу, 3 розділів, висновку, списку використаних джерел із 23 найменувань, додатків. Загальний обсяг роботи – 145 сторінок, у тому числі 59 сторінок основного тексту, 3 сторінки списку використаних джерел, 87 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці програмного додатку підтримки діяльності волонтерської організації.

В роботі проведено аналіз предметної області, проектування програмного додатку, розробку програмного додатку.

У роботі виконано огляд останніх досліджень та публікацій, аналіз програмних продуктів-аналогів, функціональне моделювання програмного додатку в IDEF0, моделювання варіантів використання та проектування бази даних.

У роботі було продемонстровано архітектуру програмного додатку, реалізацію бази даних, програмну реалізацію та використання програмного додатку.

Результатом проведеної роботи є створення програмного додатку підтримки діяльності волонтерської організації.

Практичне значення роботи полягає у розробці програмного додатку для волонтерської організації для автоматизації управління процесами.

Ключові слова: програмний додаток, волонтерська організація, база даних, внутрішньо переміщені особи, управління процесами, C#, MySQL.

ЗМІСТ

ВСТУП

Реалії сьогодення, з якими стикнулась Україна та українське суспільство, сприяли згуртуванню населення та фахівців різноманітних сфер. Не виключенням стала сфера соціальної роботи. Щоб підтримати збройні сили України, населення, яке постраждало від агресивних дій окупантів, дітей, людей похилого віку, багато українців почали займатись волонтерською діяльністю [1]. Значна кількість таких волонтерів займається збором коштів, закупкою необхідних товарів, ліків, продуктів харчування, допомагають з перевезенням та евакуацією, допомагають отримати фахову медичну допомогу, допомагають переселенцям тощо [2]. За незначний проміжок часу волонтерська діяльність в Україні перетворилось на потужний національний громадянський рух, як окремих осіб, так і громадських організацій, які спрямовують свою діяльність на надання різнопланової допомоги за окремими напрямами, а саме: матеріальної, медичної, правозахисної, психологічної різним категоріям населення. Волонтерський рух, який діє сьогодні в Україні не має аналогів.

В Україні можна помітити тренд на збільшення числа людей, які звертаються за допомогою до волонтерів [3]. З цього можна зробити висновок, що кожного дня «база» таких звернень зростає і потребує оцифрування. Звичайно можна тримати всі записи у паперовому вигляді, але це не практично в наш час, в час діджиталізації.

Отже, з метою автоматизації управління процесів у волонтерських організаціях було вирішено розробити десктоп-додаток. Проведено огляд існуючих програм за даною тематикою, виявлено їх переваги і недоліки, які будуть використані в процесі розробки даного додатку.

Метою бакалаврської роботи є створення десктоп-додатку для автоматизації управління процесами у волонтерській організації.

Для досягнення цієї мети потрібно:

– провести дослідження предметної області та аналіз прграмних засобів, що можуть використовуватися для підтримки діяльності волонтерських центрів;

– розробити технічне завдання на власну розробку та виконати планування робіт;

– виконати моделювання процесів роботи майбутнього додатку та бази даних для збереження необхідної інформації;

– розробити десктоп-додаток з базою даних для підтримки роботи програми;

– провести тестування десктоп-додатку.

За результатами представленої роботи було опубліковано тези доповіді на конференції «ІМА – 2023» [4].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

Цифровізація дозволяє впроваджувати нові інструменти, розширює можливості бізнесу, покращує взаємодію між компанією та клієнтом. Десктоп-додатки – це програми, які потребують ОС настільного комп'ютера для своєї роботи [5]. Вони встановлюються в систему через спеціальний інсталятор, використовують для роботи ресурси комп'ютера. Головна особливість таких програм – можливість працювати автономно без підключення до інтернету. Сучасні додатки все одно пропонують працювати з підключенням, але це потрібно для того, аби використовувати різні пристрої, включити в проект інших користувачів або оновлювати програму без ручного перевстановлення. Багато компаній дбають про безпеку, тому не хочуть, щоб додатки підключалися до Інтернету. У такому разі підключення можливе у закритій локальній мережі.

Вважається, що такий тип програм швидший, а функціональність набагато ширша. Розробляючи десктоп-додаток, ви отримуєте унікальні функції для бізнесу, які допоможуть вашій компанії [6]. Головна риса – широкі можливості для функціоналу. Можна реалізувати практично будь-яку ідею. При цьому інтерфейс буде зручним, звичним та інтуїтивно зрозумілим для користувачів.

Розробка десктоп-додатків для бізнесу відкриває нові горизонти [7]. Впровадження діджитал продуктів у компанію дозволяє підвищити ефективність, автоматизувати багато процесів компанії та оптимізувати роботу. Все це відбувається завдяки можливостям, які надає цифровий продукт.

Основна мета цієї системи полягає в тому, щоб зменшити робоче навантаження волонтера і мінімізувати помилки введення записів, автоматизувавши цей процес [8]. Волонтери отримують можливість заощадити час, використовуючи дану платформу. Завдяки таким перевагам

волонтери можуть користуватися кращими послугами, вносячи зміни в базу даних в зручний для них час. Це допоможе не тільки скоротити час на обробку даних.

На сьогоднішній час для волонтера існує два основних способи ведення документації. Перший – це безпосередньо ведення паперової документації. Другий – оцифровка даних за допомогою програмних додатків [9].

Порівняльна характеристика підходів до ведення документації в волонтерських організаціях наведена у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика

Назва	**Переваги**	**Недоліки**
Паперова документація	Не потребує розробки додаткового софту. Менша вірогідність втрати даних.	Складний процес доступу до даних та їх сортування. Незручність у веденні та аналізу інформації.
Десктоп-додаток	Можливість аналізу даних та зручність доступу до інформації. Відсутність паперової документації. Швидкий аналіз інформації.	Витрати на розробку та підтримку додатку. Постійна наявність інтернету для коректної роботи програми.

Як ми бачимо, використання десктоп-додатків є найзручнішим способом для ведення документації у волонтерських організаціях.

Висновки з останніх досліджень та публікацій підтверджують важливість та актуальність створення даного додатку. Переваги цифрової інформації є занадто очевидними аніж паперова документація [10].

1.2 Аналіз програмних продуктів-аналогів

У мережі інтернет наявні подібні програмні додатки, проте значна частина з них залишається недоробленою: інтерфейс може бути не зручним або застарілим, а функції не актуальними для волонтерської організації [11]. Для успішного функціонування десктоп-додатку необхідно надати волонтерам повний функціонал і зручний інтерфейс. Для визначення вимог до майбутнього програмного продукту було проведено дослідження аналогів, таких як «Excel» та «Dilovod».

Головна сторінка додатку «Excel» представлена на рисунку 1.1.

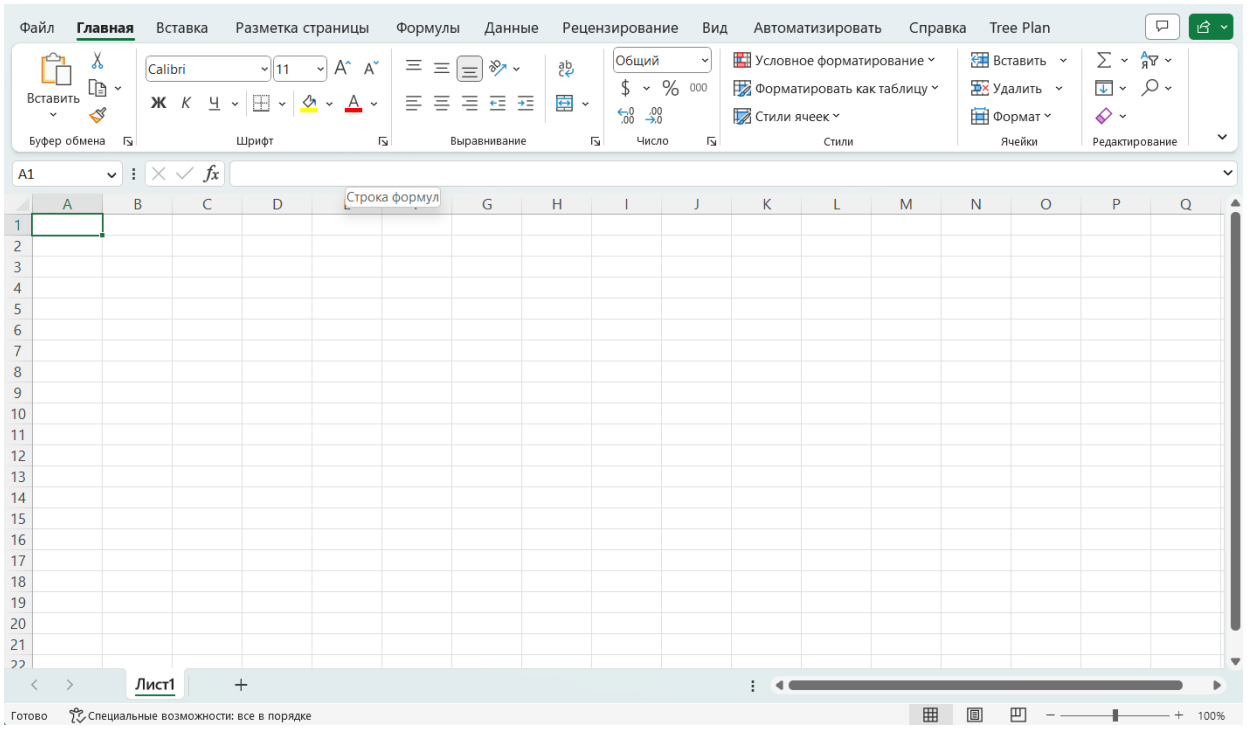


Рисунок 1.1 – Головна сторінка десктоп-додатку «Excel»

На сторінці головній додатку «Excel» розміщено забагато зайвих функцій, які не є доцільними і точно не будуть використовуватися в роботі. Дизайн сторінки виглядає не сучасно і не практично [12].

Ще один альтернативний програмний додаток - «Dilovod» [21], головна сторінка додатку представлена на рисунку 1.2.

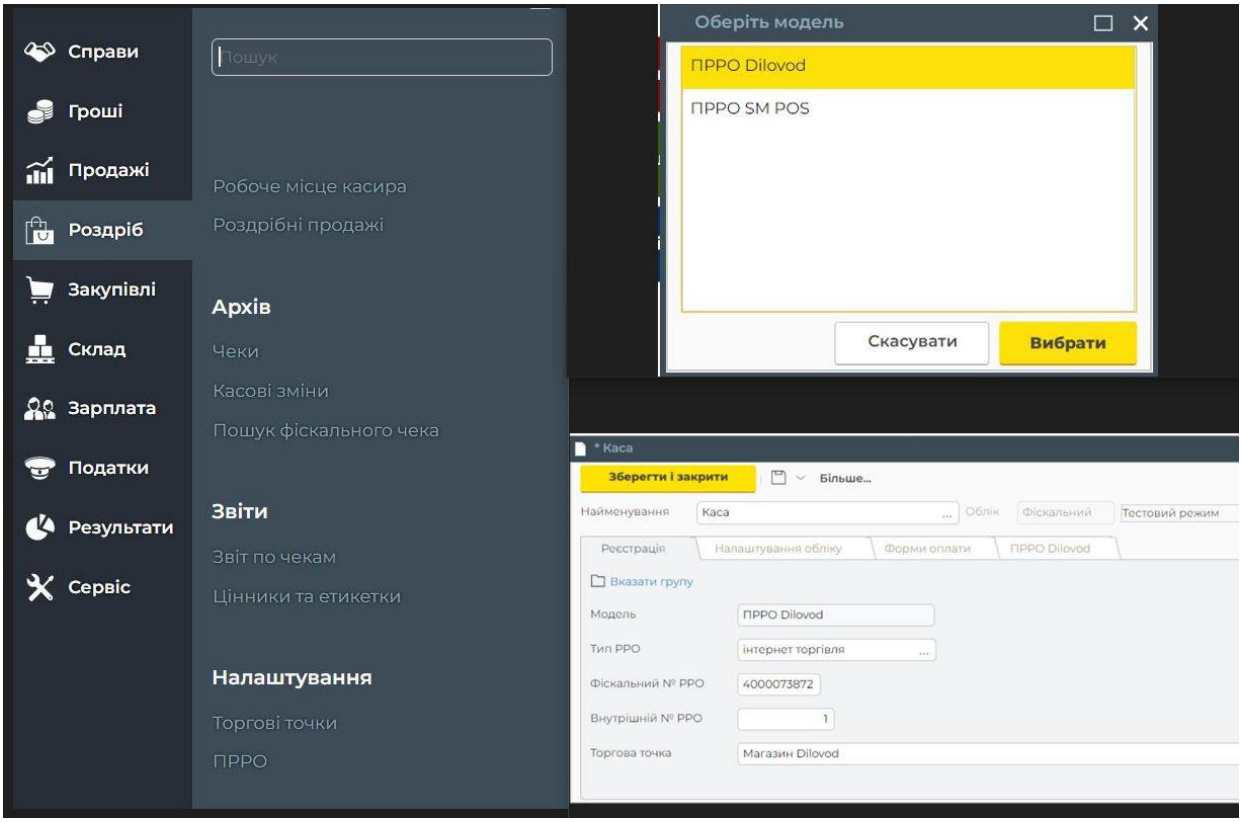


Рисунок 1.2 – Головна сторінка десктоп-додатку «Dilovod»

Даний аналог має сучасний дизайн та зручний інтерфейс. Всі функції логічно пов’язані між собою. Навігація по основним функціям додатку побудована досить гарно, що є величезним плюсом у разі використання. Але повністю відсутня можливість розмежування прав доступу, враховуючи що в нашій системі повинні бути мінімум два актори: адміністратор та волонтер. Також додаток є повністю платним, тому за використання та доступ до функціоналу треба буде заплатити [13].

У результаті проведеного аналізу аналогу можна зробити підсумки, які представлено в таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця характеристик аналогів

Характеристика/ Додаток	«Excel»	«Dilovod»
Сучасний дизайн	-	+
Зручний інтерфейс	-	+
Адаптивність	-	+
Функціональність	-	-
Навігація	-	+
Розмежування прав доступу	+	-
Безкоштовне використання	+	-

З таблиці 1.2 можна зробити висновок, що наявний додаток не підходить для вирішення поставленої задачі, тому було вирішено розробити власний десктоп-додаток, щоб виправити всі недоліки, та додати саме ті функції, які потрібні для роботи нашого додатку. Розроблюваний програмний продукт повинен мати сучасний дизайн, зручну навігацію та інтерфейс.

1.3 Постановка задачі

Основною метою проекту є розробка десктоп-додатку для підтримки діяльності волонтерського центру зі зручним та інтуїтивно зрозумілим інтерфейсом.

Використання даного додатку дозволить волонтерам заощадити час на пошук та обробку інформації. Додаток надасть можливість волонтерам швидко додавати, знаходити потрібну інформацію, а також аналізувати її та вести певну звітність.

Необхідно виконати ряд завдань, щоб досягти поставленої мети:

− створити модель та структуру десктоп-додатку;

- вибрати технології розробки;
- створити прототип десктоп-додатку;
- реалізувати структуру та функціонал десктоп-додатку;
- виконати тестування;
- запропонувати можливі шляхи покращення для подальшої роботи десктоп-додатку.

Технічне завдання на розробку продукту у повному обсязі наведено у додатку А.

Десктоп-додаток буде написано із використанням мови програмування C# [14]. Для управління та зберігання даних в додатку буде використовуватися реляційна база даних – MySQL [15].

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1 Функціональне моделювання програмного додатку в IDEF0

IDEF0 представляє собою графічну нотацію. Основним призначенням нотації є формалізація та опис бізнес-процесів. Метою методики є побудова функціональної схеми, яка детально описує процеси з точністю, яка є достатньою для повного моделювання системи [22].

Функціональне моделювання процесу підтримки діяльності волонтера в нотації IDEF0 представлена на рисунку 2.1.

Рисунок 2.1 – Функціональна діаграма IDEF0

На вхід поступають дані про волонтера, про замовника допомоги та про пакунок. Реалізація основного процесу відбуватиметься за рахунок використання програмного додатку, що розробляється. Обмежують даний процес нормативні документи. Окрім цього задіяні ще такі механізми реалізації, як технічні засоби, системне програмне забезпечення та програмний додаток. На виході маємо інформацію про оформлену видачу.

Наступний етап – це декомпозиція контексту, що ґрунтується на тому, що комплексний процес ділять на складові. Декомпозиція є основним поняттям стандарту IDEF0. Принцип декомпозиції полягає у розбитті складного процесу на його менші складові частини. Модель системи представляється у вигляді ієрархічної структури діаграм [22].

Декомпозиція 1-го рівня моделювання процесу підтримки діяльності волонтерської організації у нотації IDEF0 представлена на рисунку 2.2.

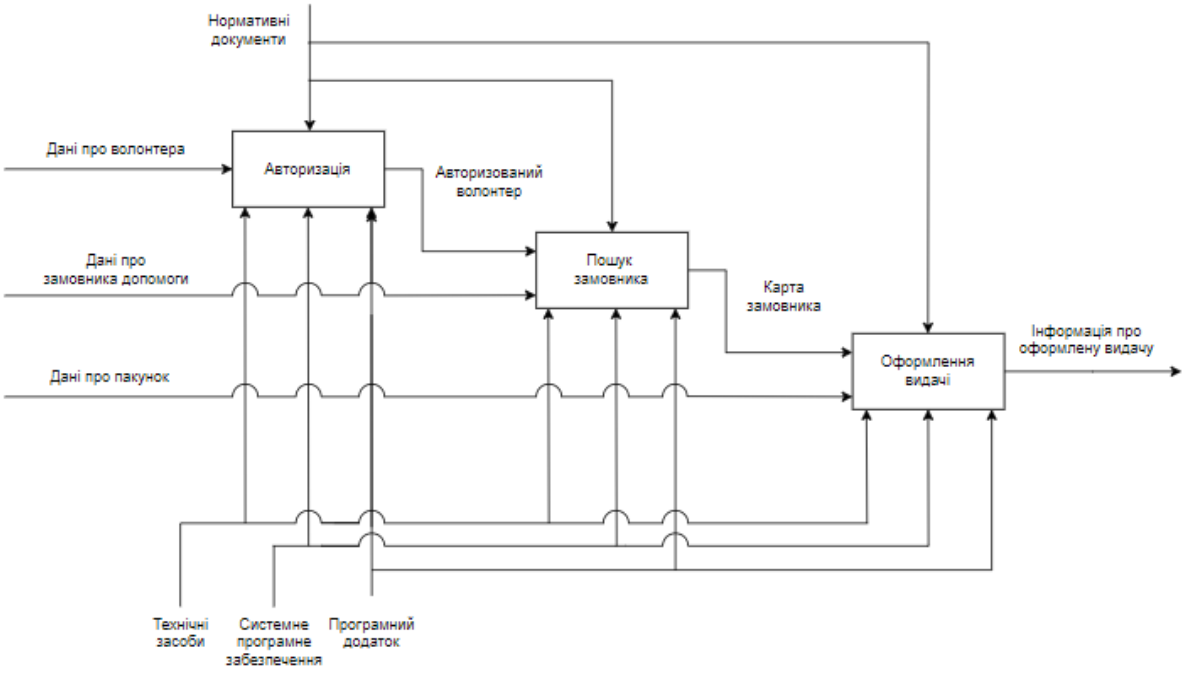


Рисунок 2.2 – Декомпозиція функціональної моделі

Процес підтримки діяльності волонтера складається з таких етапів, як авторизація, пошук замовника, оформлення видачі. Авторизація – це найперший етап, який заключається у вході в систему волонтера під своїм унікальним іменем. Пошук замовника – це другий етап, який заключається в пошуку замовника за його даними. І останній етап – це оформлення видачі, який супроводжує фізичну дію видачі замовленя, а в системі полягає у формуванні відповідного запису в базі даних.

2.2 Діаграма варіантів використання

В UML, діаграми варіантів використання моделюють поведінку системи та визначають системні вимоги системи. Діаграми варіантів використання описують функції високого рівня та область застосування системи. Ці діаграми також показують взаємодію між системою та її акторами. Варіанти використання та актори на діаграмах варіантів використання описують, що робить система та як актори її використовують, але не те, як система працює всередині. [23]

Діаграма варіантів використання в нотації UML представлена на рисунку 2.3.

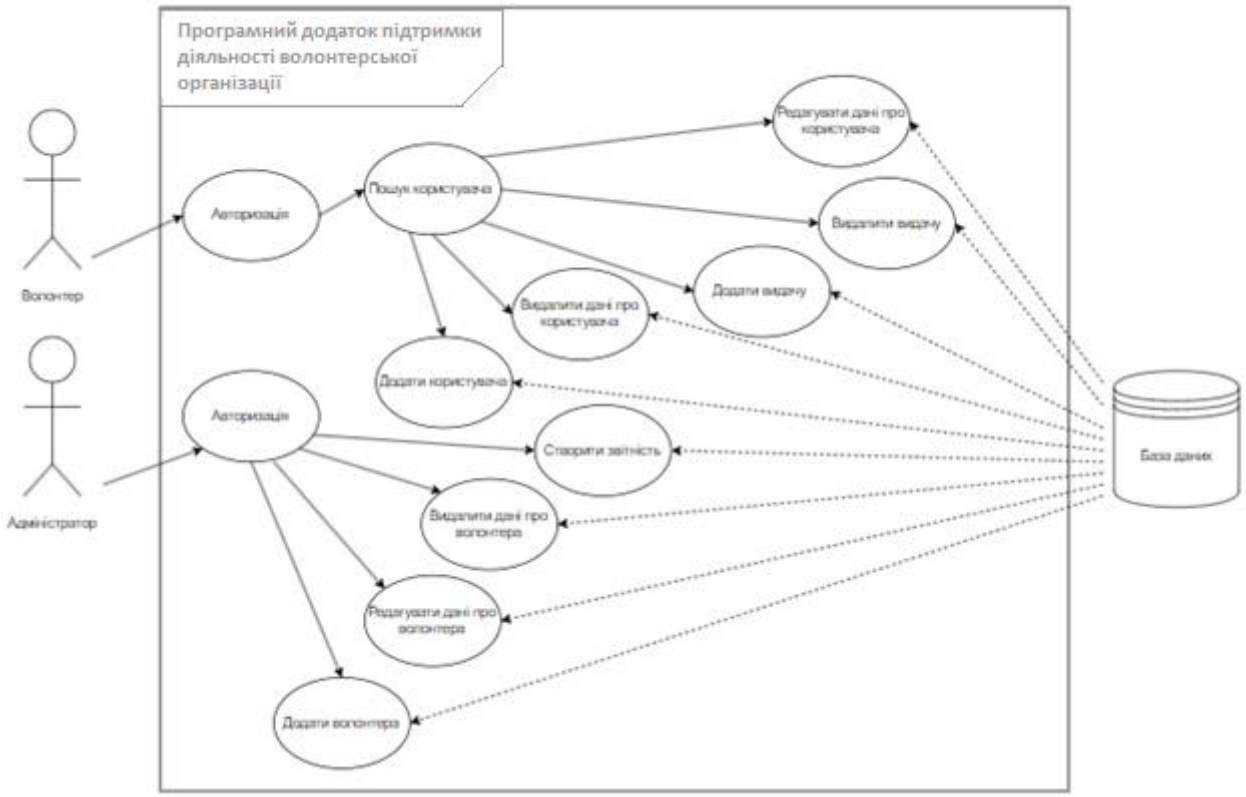


Рисунок 2.3 – Діаграма варіантів використання

У даному десктоп-додатку існує 3 типи акторів:

− Волонтер: зареєстрований волонтер, який може додавати, редагувати, видаляти дані про замовників та додавати, видаляти видачі;

− Адміністратор: підтримує роботу даного десктоп-додатку, а саме може додавати, редагувати, видаляти дані про волонтера та створювати звітність;

− База даних: реагує на запити та зміни даних.

2.3 Проектування бази даних

Після створення шаблонів сторінок десктоп-додатку, необхідно не тільки наповнити їх коректною інформацією й налагодити роботу з нею, але й забезпечити її правильне зберігання. Для описання фізичної структури БД використовуються фізичні моделі даних (рис. 2.4).

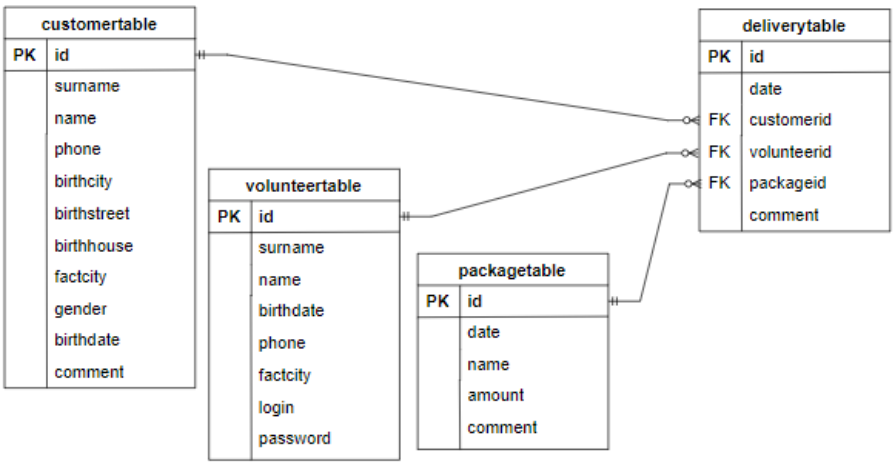


Рисунок 2.4 –Модель бази даних

Таблиця customertable створена для зберігання даних про замовників допомоги. Має первинний ключ id та інші поля, а саме прізвище, ім’я, номер телефону, дату народження, повну адресу проживання, фактичне місто проживання, стать та коментар.

Таблиця voluntteertable створена для зберігання даних про волонтерів. Має первинний ключ id та інші поля, а саме прізвище, ім’я, номер телефону, дату народження, логін та пароль.

Таблиця packagetable створена для зберігання даних про пакунки. Має первинний ключ id та інші поля, а саме дату додавання пакунку, назву пакунку, кількість пакунків та коментар.

Таблиця deliverytable створена для зберігання даних про замовників допомоги. Має первинний ключ id та 3 зовнішніх ключа, а саме customerid, volunteerid та packageid. Також є інші поля, а саме дату видачі та коментар.

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Архітектура додатку

Для розробки архітектури даного програмного додатку перш необхідно визначити взаємодії між волонтером та його основними компонентами за допомогою High Level Design (HLD) діаграми.

Архітектура програмного додатку підтримки діяльності волонтерської організації складається з наступних елементів:

− обробник відображення (Handler Mapping), який на основі запиту волонтера обирає необхідний контролер для подальшої обробки;

− контролер (Controller), який взаємодіє з сервісами для відправлення волонтеру коректної генерації вигляду;

− вигляд (View), який отримує дані від контролера для генерації відповіді волонтеру;

− сервіси (Service), який виконує роботу з даними та надсилає дані до контролеру;

− сутності (Entity), яка являється описом сутностей бази даних для використання їх в сервісі;

− репозиторії (Repository), який забезпечує доступ до бази даних;

− база даних (Database), яка забезпечує програмний додаток даними.

HLD діаграма представлена на рисунку 3.1

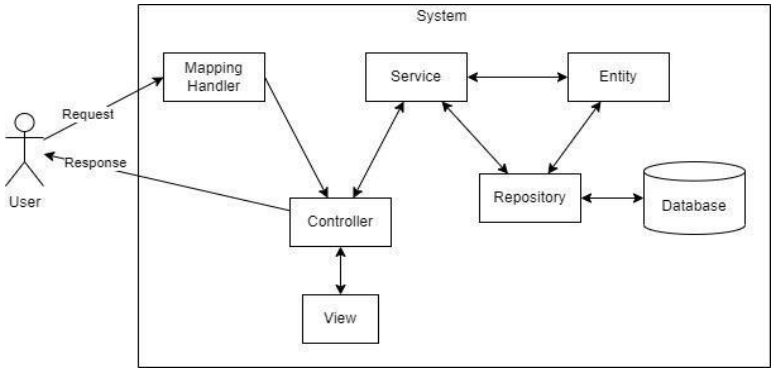


Рисунок 3.1 – HDL діаграма

3.2 Реалізація бази даних

Для реалізації бази даних було використано СУБД MySQL. На прикладі таблиці volunteertable розглянемо шлях її створення.

Створення таблиці volunteertable

```
CREATE TABLE `volunteertable` (
  `id` int NOT NULL,
  `surname` varchar(30) NOT NULL,
  `name` varchar(20) NOT NULL,
  `birthdate` date NOT NULL,
  `phone` varchar(10) NOT NULL,
  `factcity` varchar(25) NOT NULL,
  `login` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL
);
```

Додавання первинного ключа

```
ALTER TABLE `volunteertable`
  ADD PRIMARY KEY (`id`);
```

Додавання значень в таблицю

```
INSERT INTO `volunteertable` (`id`, `surname`, `name`, `birthdate`, `phone`, `factcity`, `login`, `password`) VALUES
(2, 'Ткаченко', 'Настя', '1999-03-07', '502387455', 'Полтава', 'tkach', '12345');
```

Загальна схема бази даних представлена на рисунку 3.2

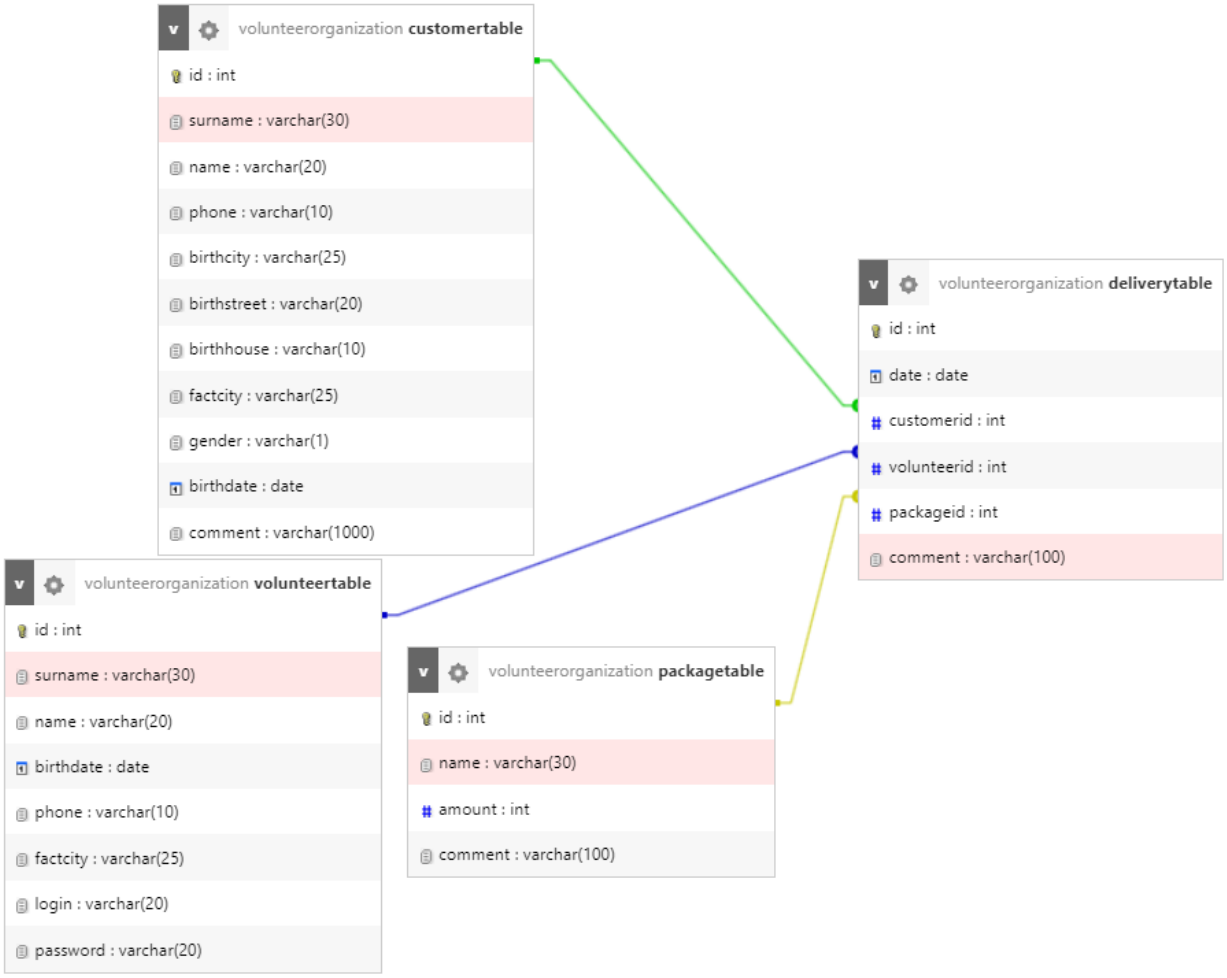


Рисунок 3.2 – Загальна схема бази даних

3.3 Програмна реалізація

Першим кроком необхідно створити клас MySQL (файл MySQL.cs, рис. 3.3). Методи файлу будуть використовуватися для роботи з базою даних. Зокрема:

- OpenConnection() – на вхід не приймає значень – перевіряє стан підключення, якщо підключення вимкнене, то вмикає його.
- CloseConnection() – на вхід не приймає значень – перевіряє стан підключення, якщо підключення ввімкнене, то вимикає його.
- GetConnection() – на вхід не приймає значень – повертає значення підключення.

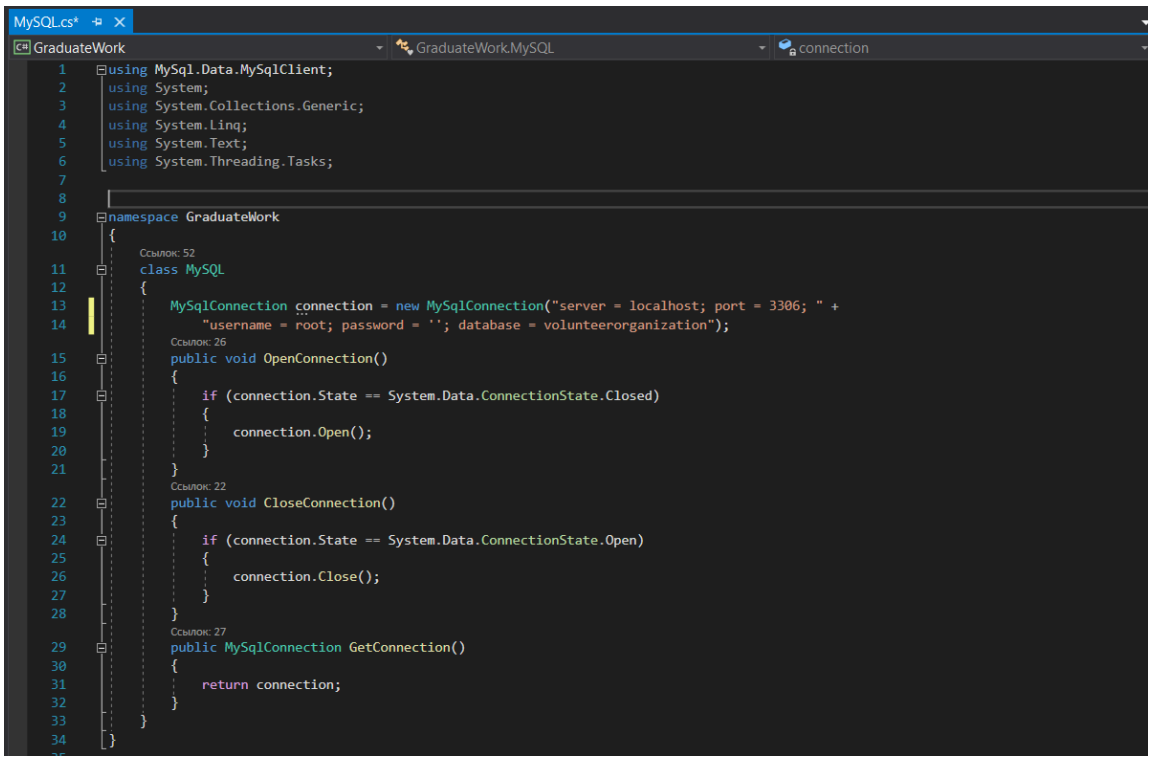


Рисунок 3.3 – MySQL.cs

Наступним кроком треба створити файл під назвою Customer.cs (рис. 3.4), він потрібен для обробки інформації про замовника.

Зокрема:

- Customer(string surname, string name, int phone, string birthcity, string birthstreet, string birthhouse, string factcity, string gender, DateTime birthdate, string comment) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.
- Customer(int id, string surname, string name, int phone, string birthcity, string factcity) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.

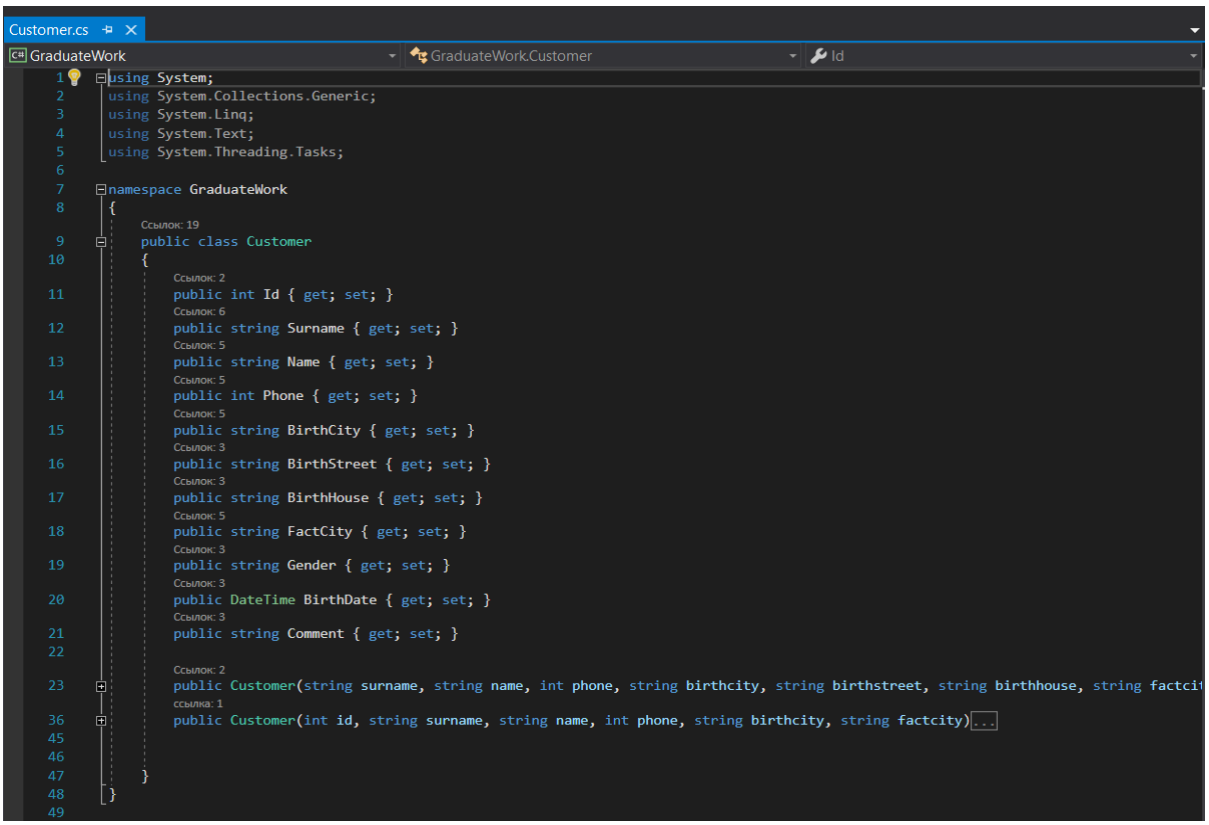


Рисунок 3.4 – Customer.cs

Наступним кроком треба створити файл під назвою Delivery.cs (рис. 3.5), він потрібен для обробки інформації про видачі.

Зокрема:

- Delivery(int id, DateTime date, string name_p, string fullname_v) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.
- Delivery(DateTime date, string name_p, string fullname_c, string phone_c, string city_c, string fullname_v) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.

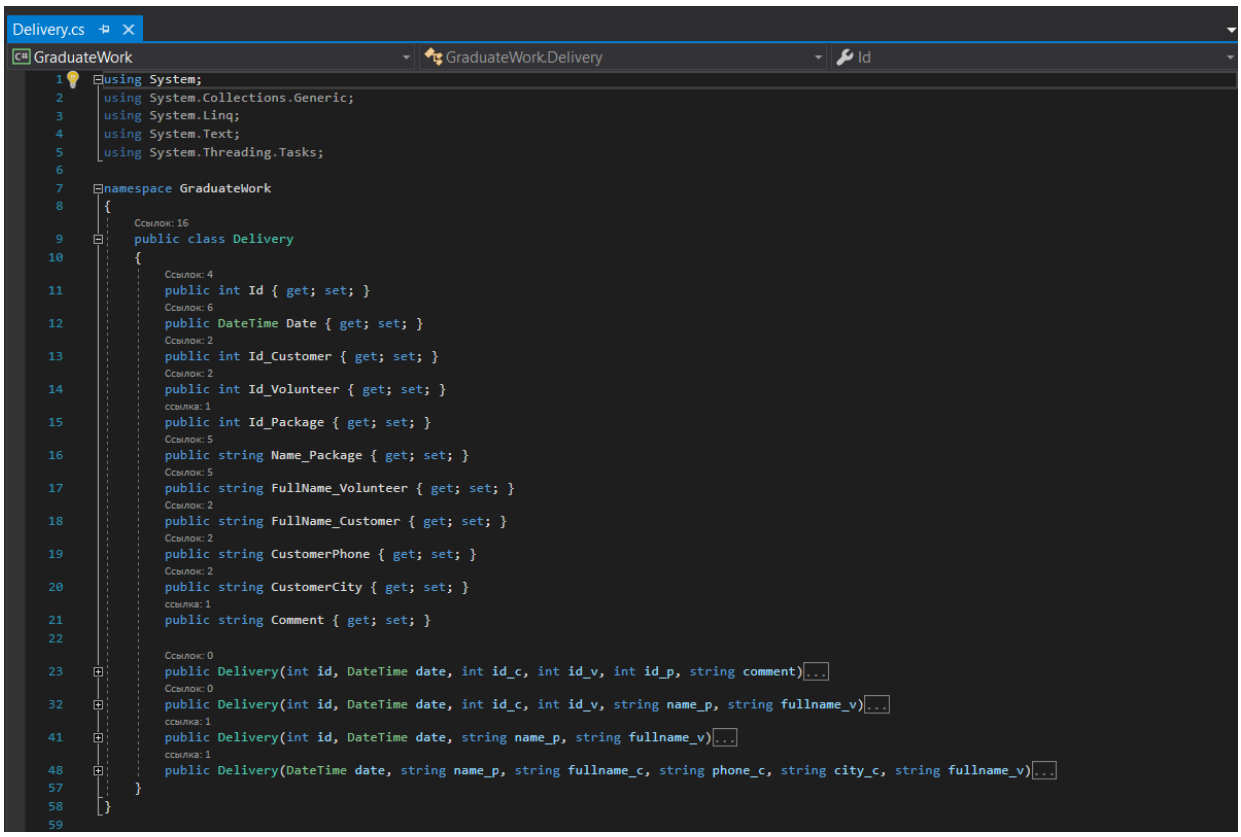


Рисунок 3.5 – Delivery.cs

Наступним кроком треба створити файл під назвою Login.cs (рис. 3.6), він потрібен для роботи з формою авторизації волонтера.

Зокрема:

- CheckUserLoginAndPassword(string _login, string _password) – на вхід приймає значення логіна та паролю для входу в систему – повертає результат перевірки коректності логіну та паролю.

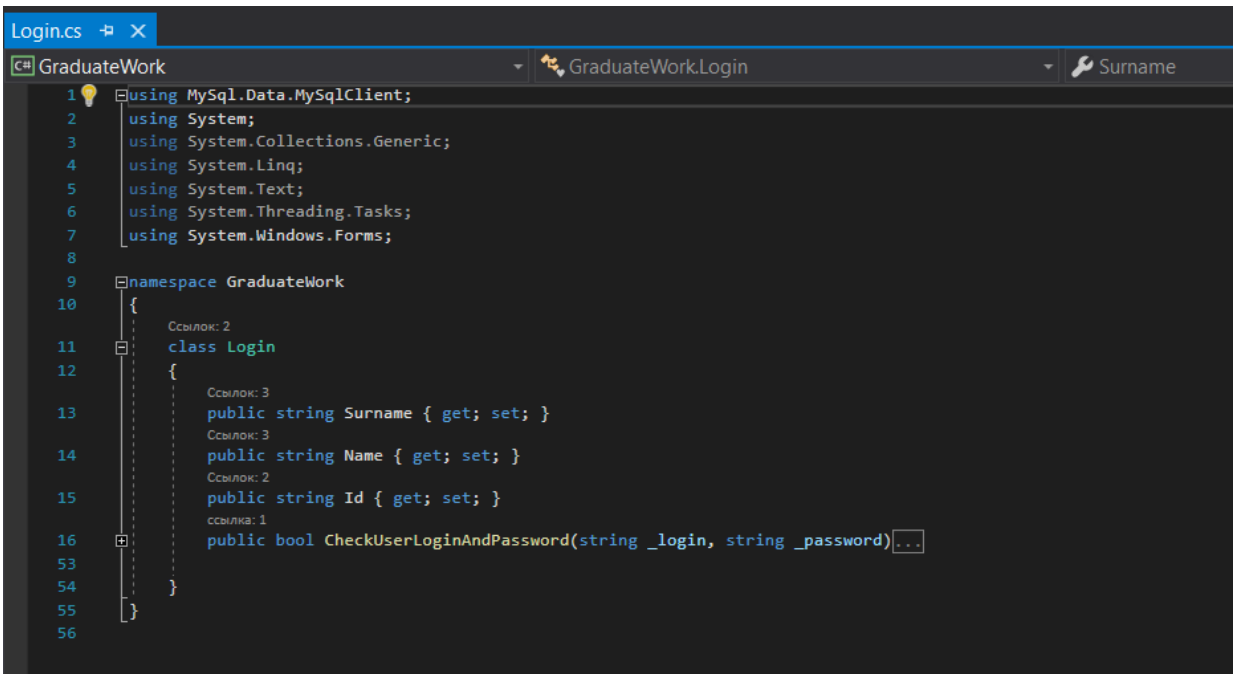


Рисунок 3.6 – Login.cs

Наступним кроком треба створити файл під назвою Package.cs (рис. 3.7), він потрібен для обробки інформації про пакунки.

Зокрема:

- Package(string id, string _n, int _a, string _c) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.

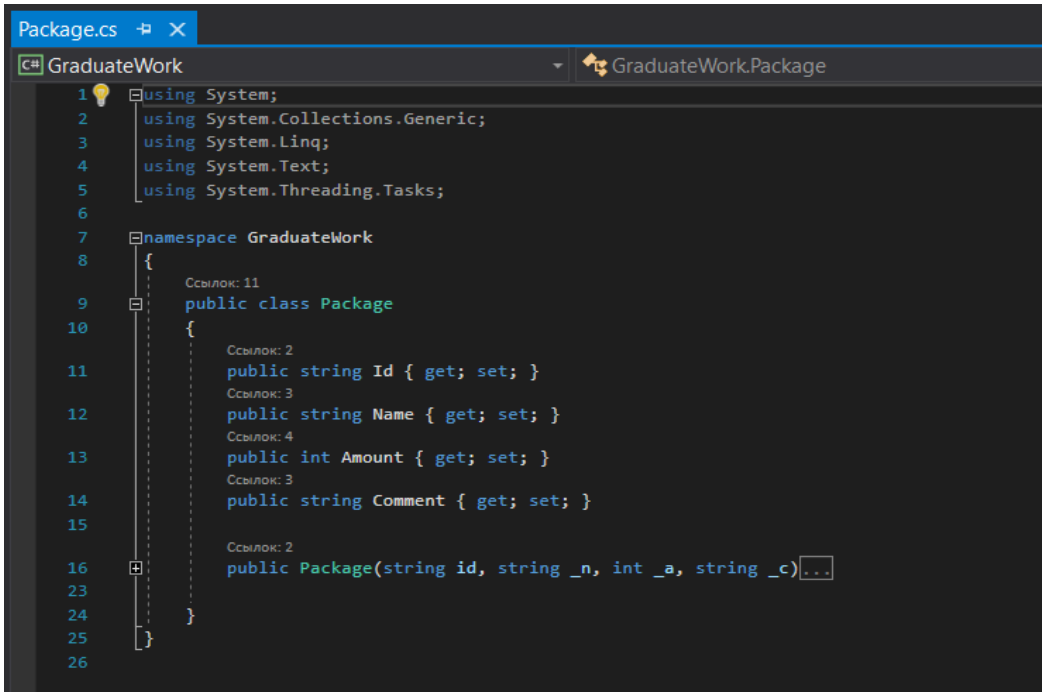


Рисунок 3.7 – Package.cs

Також потрібно створити файл під назвою Program.cs (рис. 3.8), з якого буде запускатися наш програмний додаток.

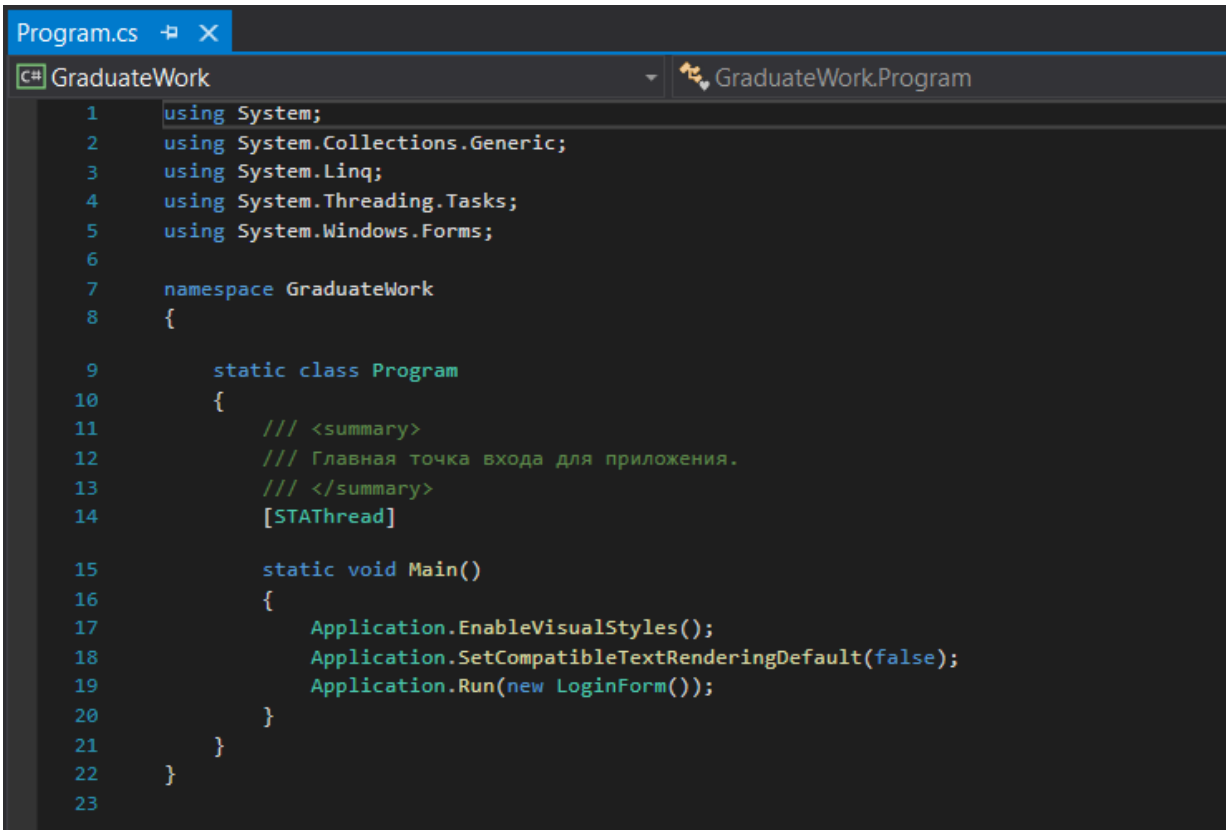


Рисунок 3.8 – Program.cs

Наступним кроком треба створити файл під назвою Volunteer.cs (рис. 3.9), він потрібен для обробки інформації про волонтера.

Зокрема:

- Volunteer(int id, string surname, string name, DateTime birthdate, int phone, string city, string login, string password, string volunteerid, string count) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.
- Volunteer(int id, string surname, string name, DateTime birthdate, int phone, string city, string login, string password) – на вхід приймає значення полів класу – задає значення змінних екземпляра, визначених у класі.

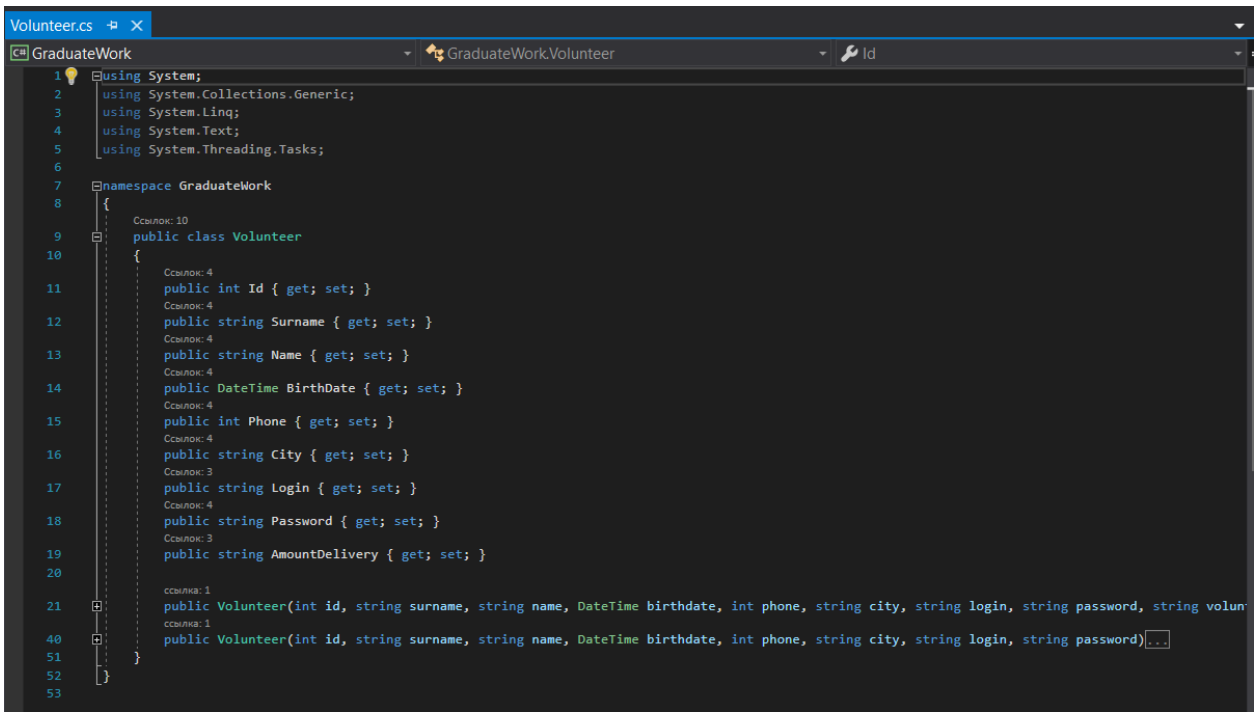


Рисунок 3.9 – Volunteer.cs

Створити форму AddCustomerForm.cs (рис. 3.10), вона потрібна для додавання замовників в систему.

Зокрема:

- buttonAdd_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додає дані нового замовника до системи.
- buttonClear_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – очищує всі поля, які були раніше введені.
- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

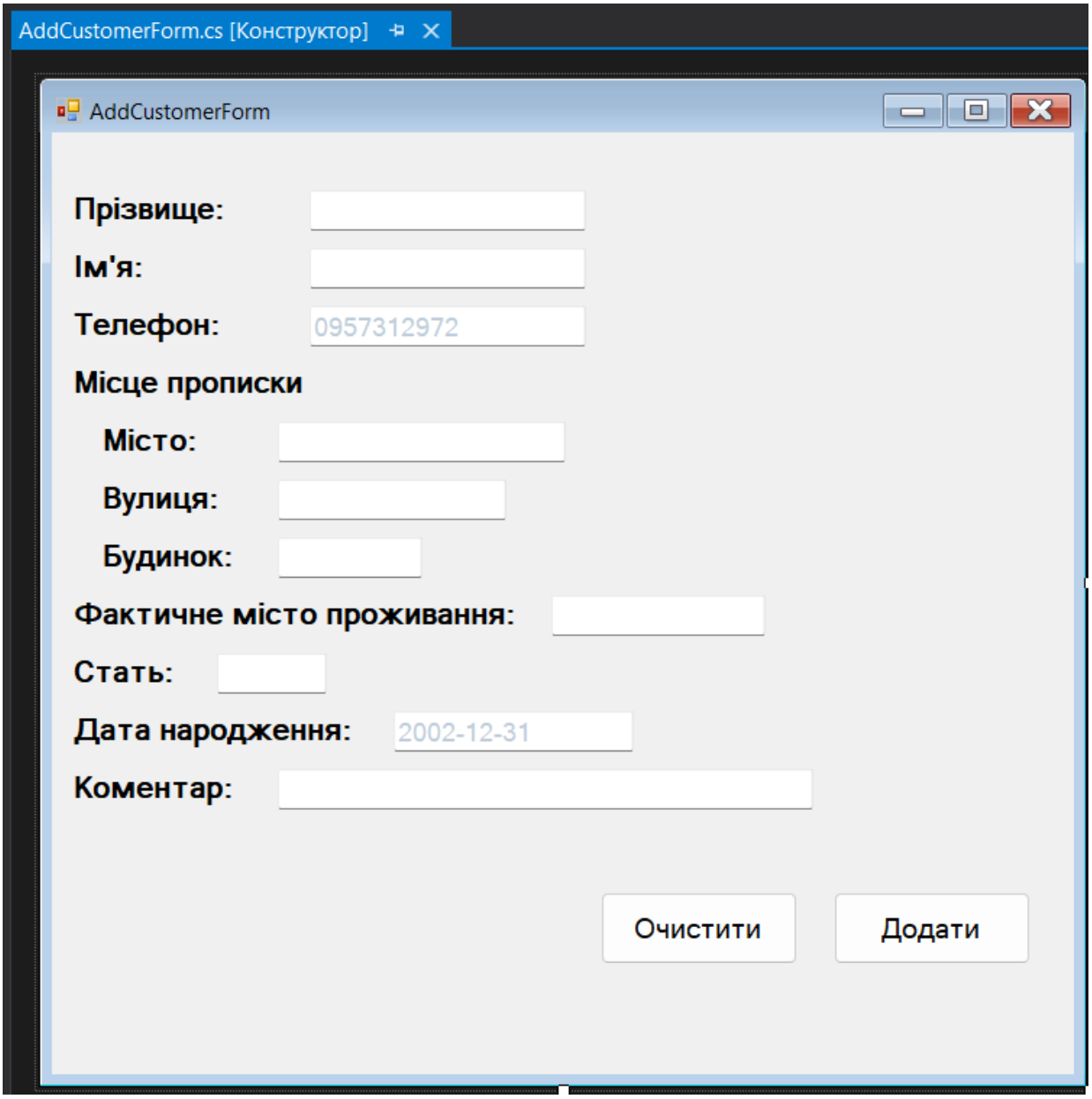


Рисунок 3.10 – AddCustomerForm.cs

Створити форму AddDeliveryForm.cs (рис. 3.11), вона потрібна для додавання видач в систему.

Зокрема:

- buttonSavePackage_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додає дані нової видачі до системи.
- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який

містить посилання на елемент керування/об'єкт, що спричинив подію – очищує всі поля, які були раніше введені.

- ComboBoxShow() – на вхід не приймає значень – відображає доступні пакунки в системі.

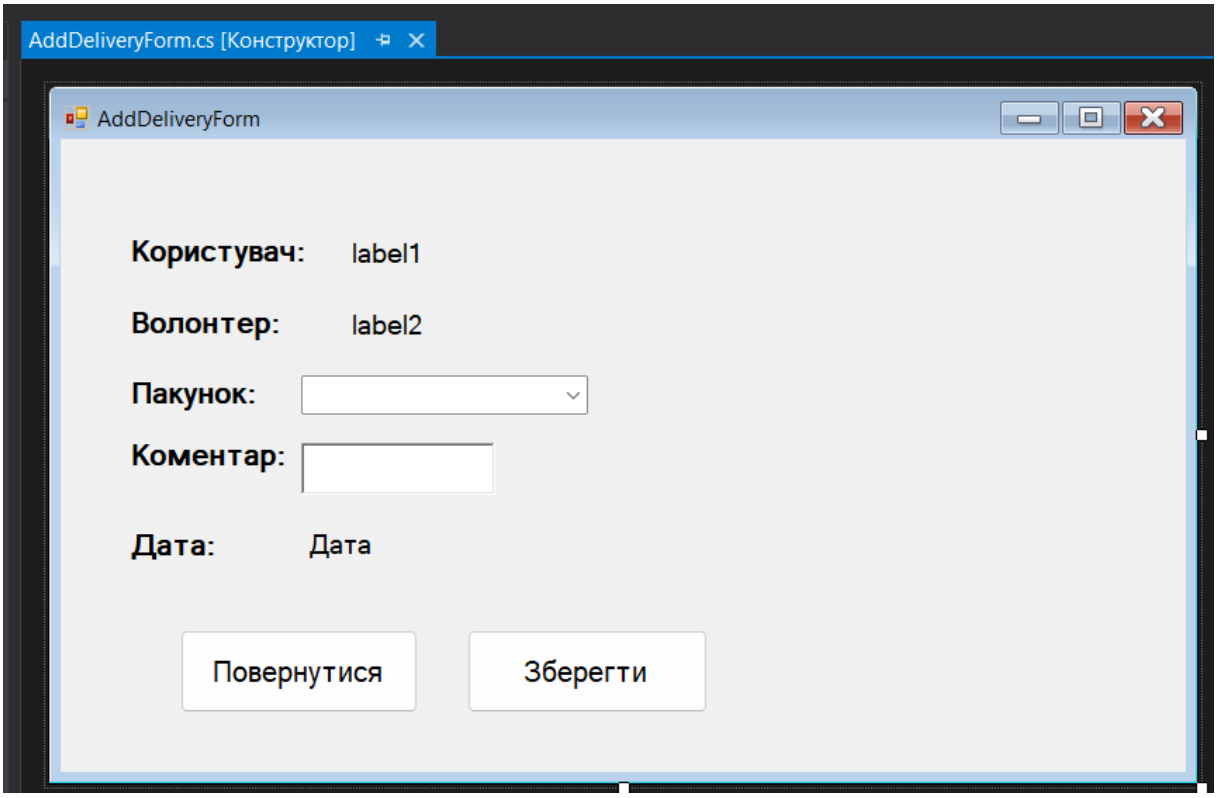


Рисунок 3.11 – AddDeliveryForm.cs

Створити форму AddPackageForm.cs (рис. 3.12), вона потрібна для додавання пакунків в систему.

Зокрема:

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додає новий пакунок до системи.
- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

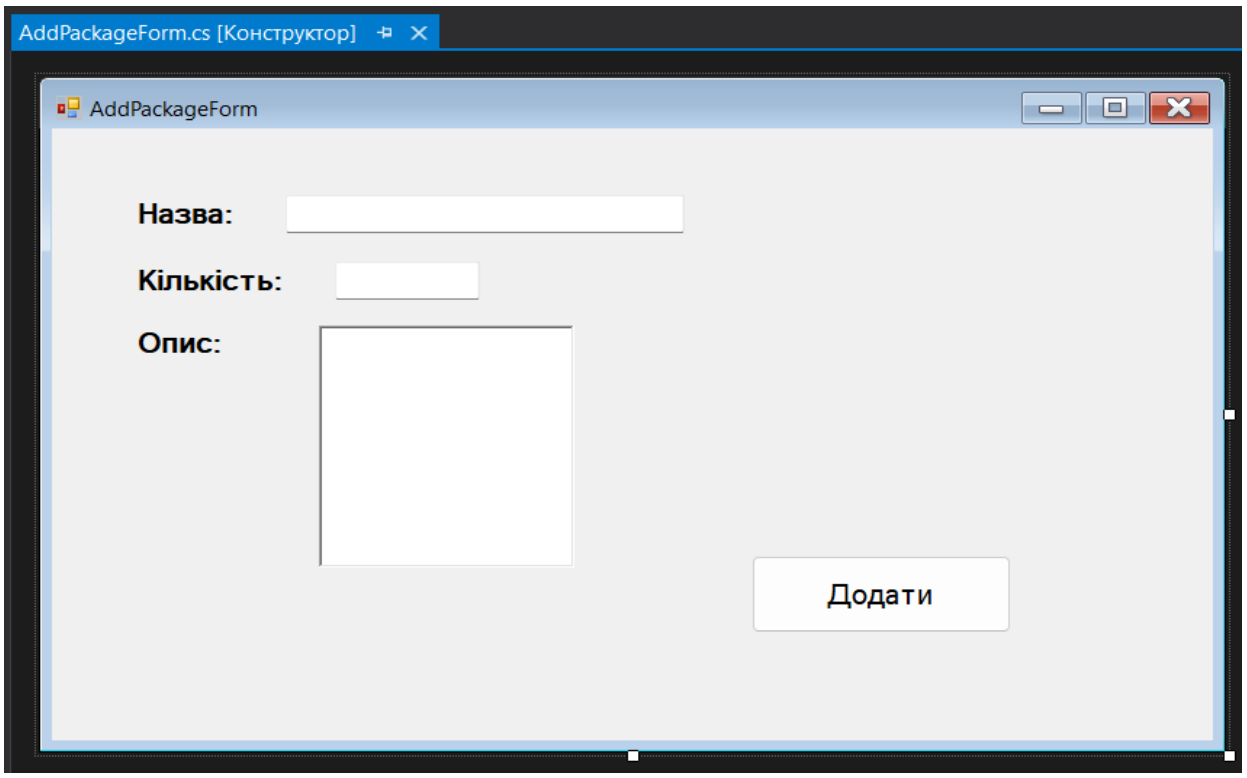


Рисунок 3.12 – AddPackageForm.cs

Створити форму AddVolunteerForm.cs (рис. 3.13), вона потрібна для додавання волонтерів в систему.

Зокрема:

- buttonAdd_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додає нового волонтера до системи.
- buttonReturn_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – закриває це вікно і повертається до попереднього вікна.
- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

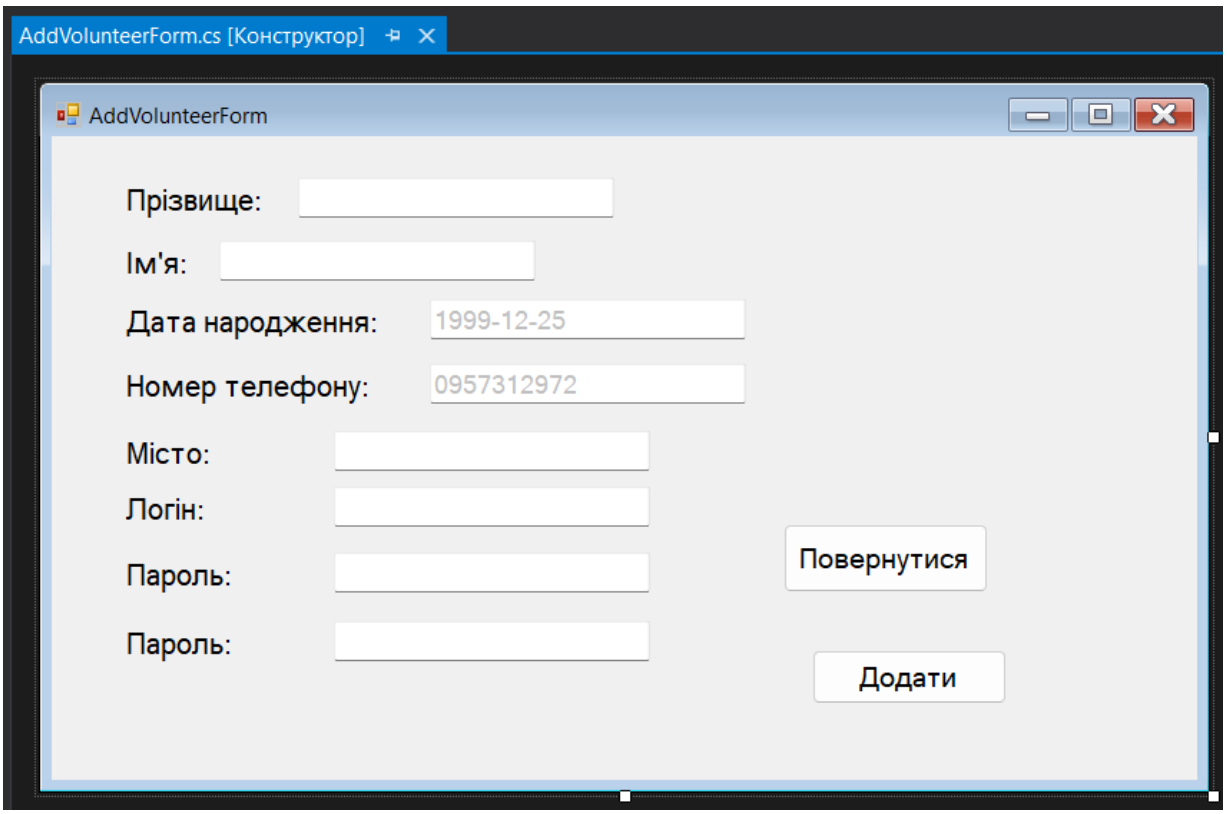


Рисунок 3.13 – AddVolunteerForm.cs

Створити форму AdminPanel.cs (рис. 3.14), вона потрібна для роботи адміністратора. Представлено 3 функції для адміністратора.

Зокрема:

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відкриває інформацію про волонтерів.
- PackageListbutton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відкриває інформацію про пакунки
- Statbutton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відкриває статистику.

-

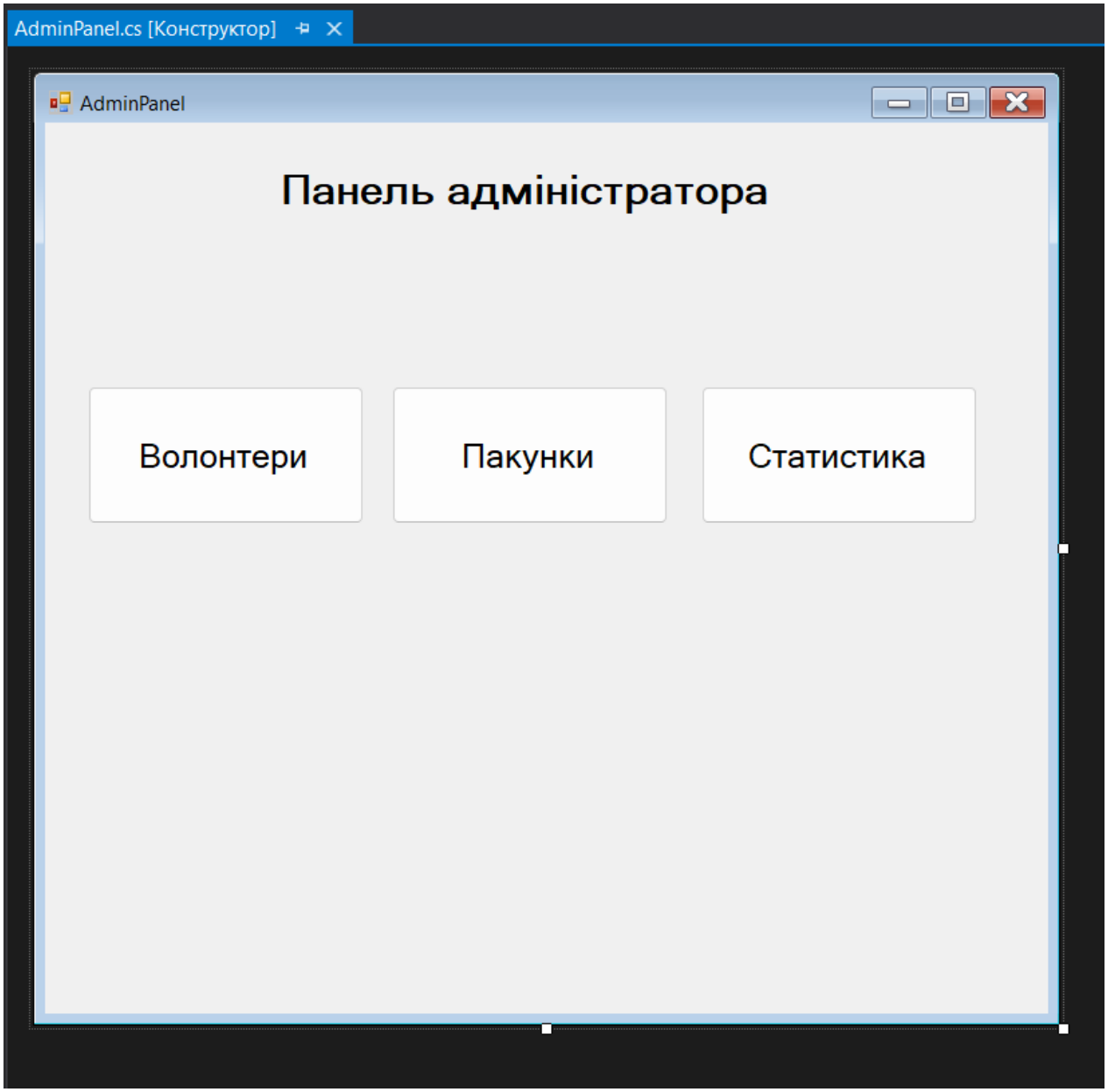


Рисунок 3.14 – AdminPanel.cs

Створити форму DeliveryForm.cs (рис. 3.15), вона потрібна для адміністратора, щоб він мав змогу переглянути та редагувати актуальні пакунки, створити по ним звіт та додати нові пакунки в систему.

Зокрема:

- button2_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – створює MS Word звіт.

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – вікриває форму додавання пакунку в систему.
- GetPack() – на вхід не приймає значень – повертає інформацію про пакунки в системі.

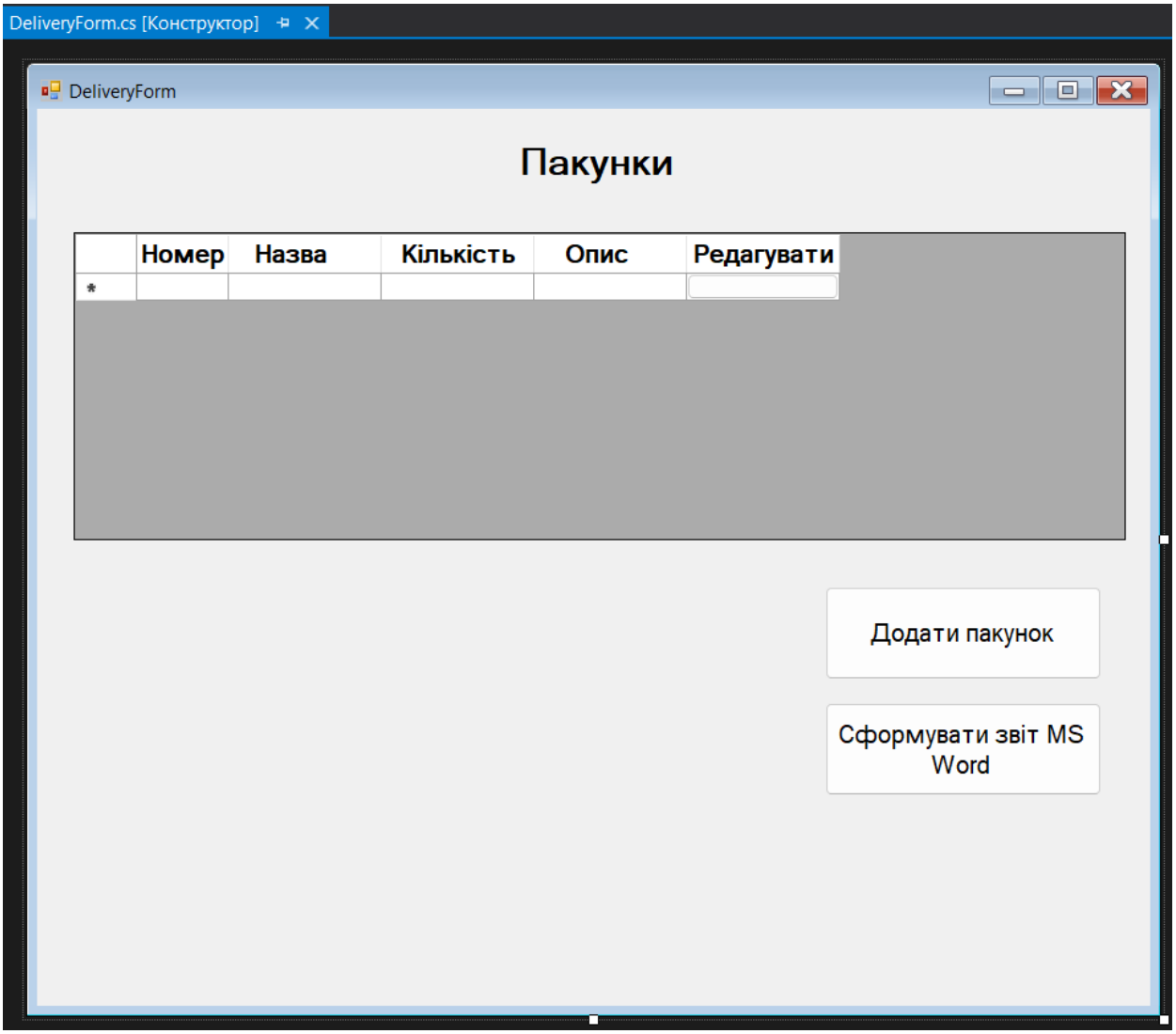


Рисунок 3.15 – DeliveryForm.cs

Створити форму DeliveryStatForm.cs (рис. 3.16), вона потрібна для відображення статистики по видачам(за конкретну дату та/або вид пакунку) та формування звіту по видачам.

Зокрема:

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відображає інформацію про видані пакунки.
- button2_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – створює MS Word звіт.

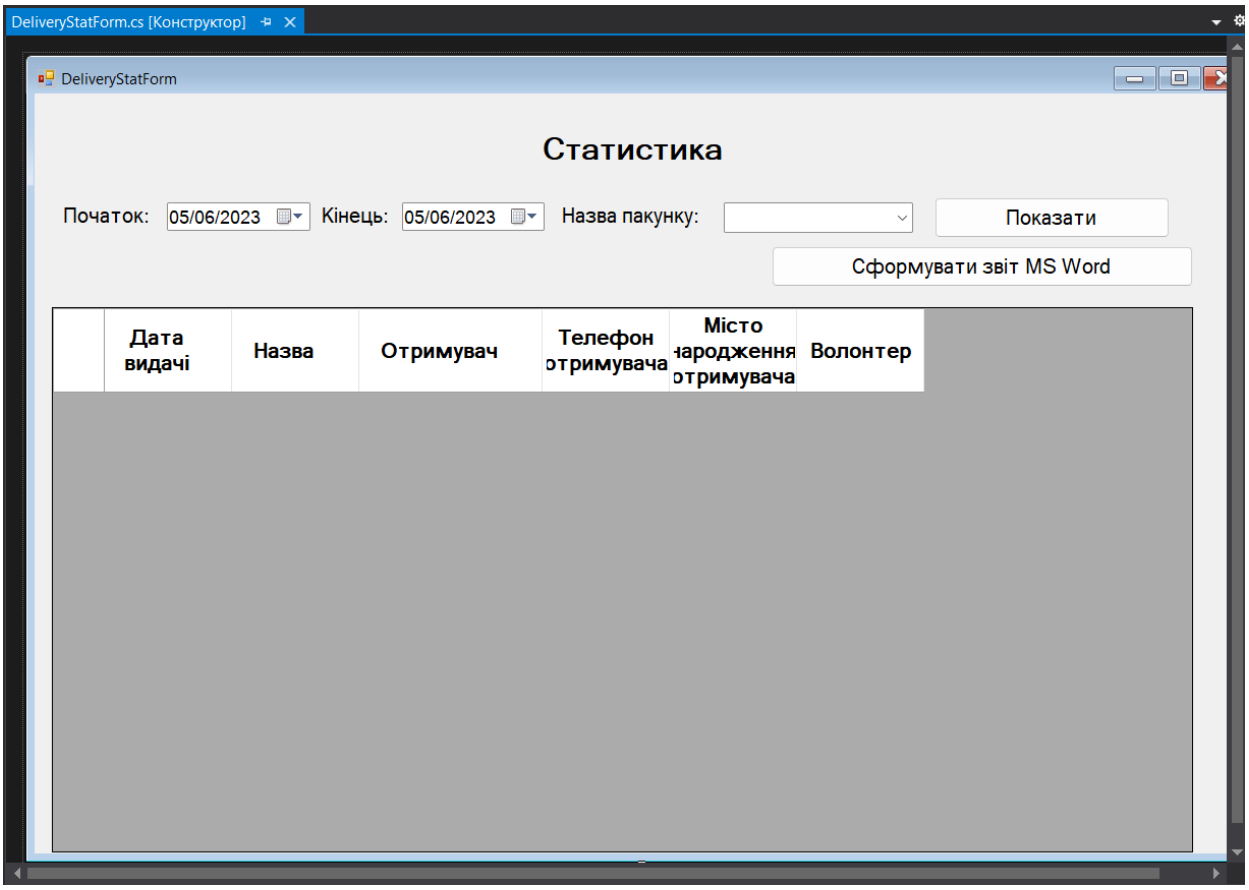


Рисунок 3.16 – DeliveryStatForm.cs

Створити форму EditCustomerForm.cs (рис. 3.17), вона потрібна для редагування даних про замовників в системі.

Зокрема:

- RefreshCustomerButton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – оновлює дані про замовника.
- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – закриває це вікно і повертається до попереднього вікна.
- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

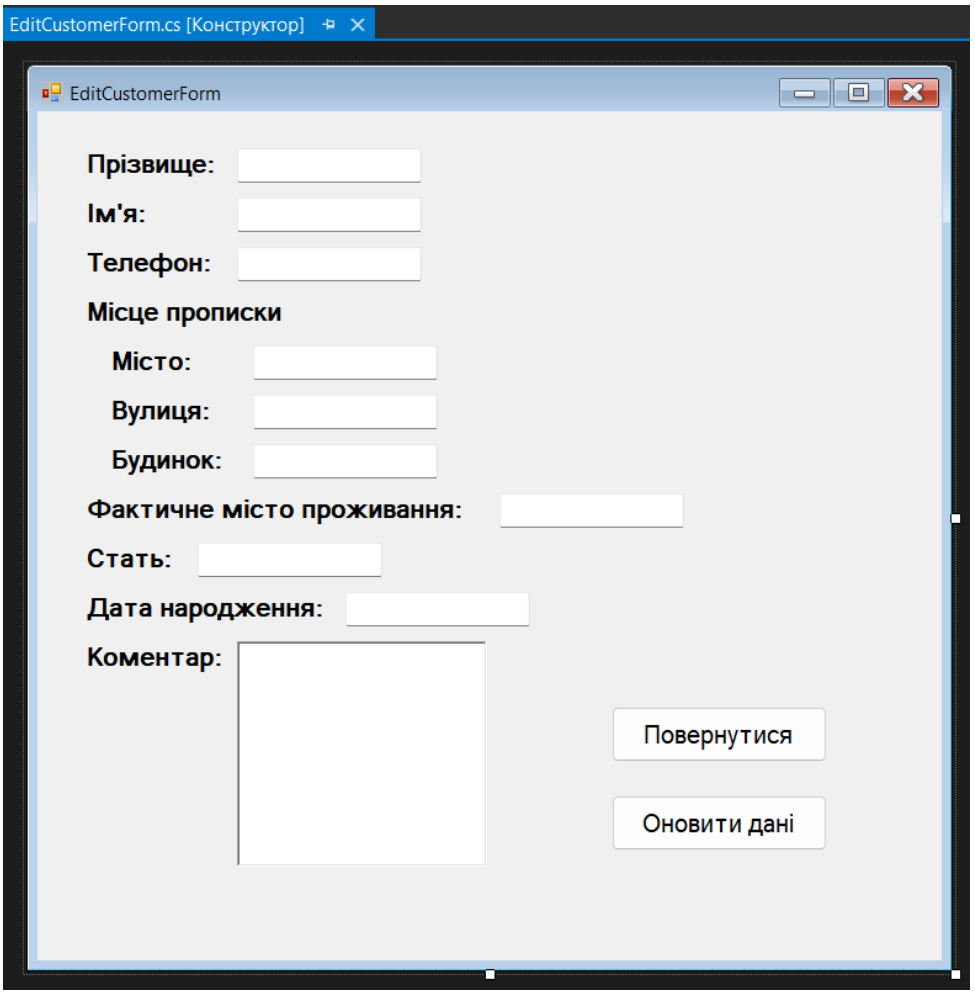


Рисунок 3.17 – EditCustomerForm.cs

Створити форму EditVolunteerForm.cs (рис. 3.18), вона потрібна для редагування даних про волонтерів в системі.

Зокрема:

- buttonAdd_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – оновлює дані про волонтера.
- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

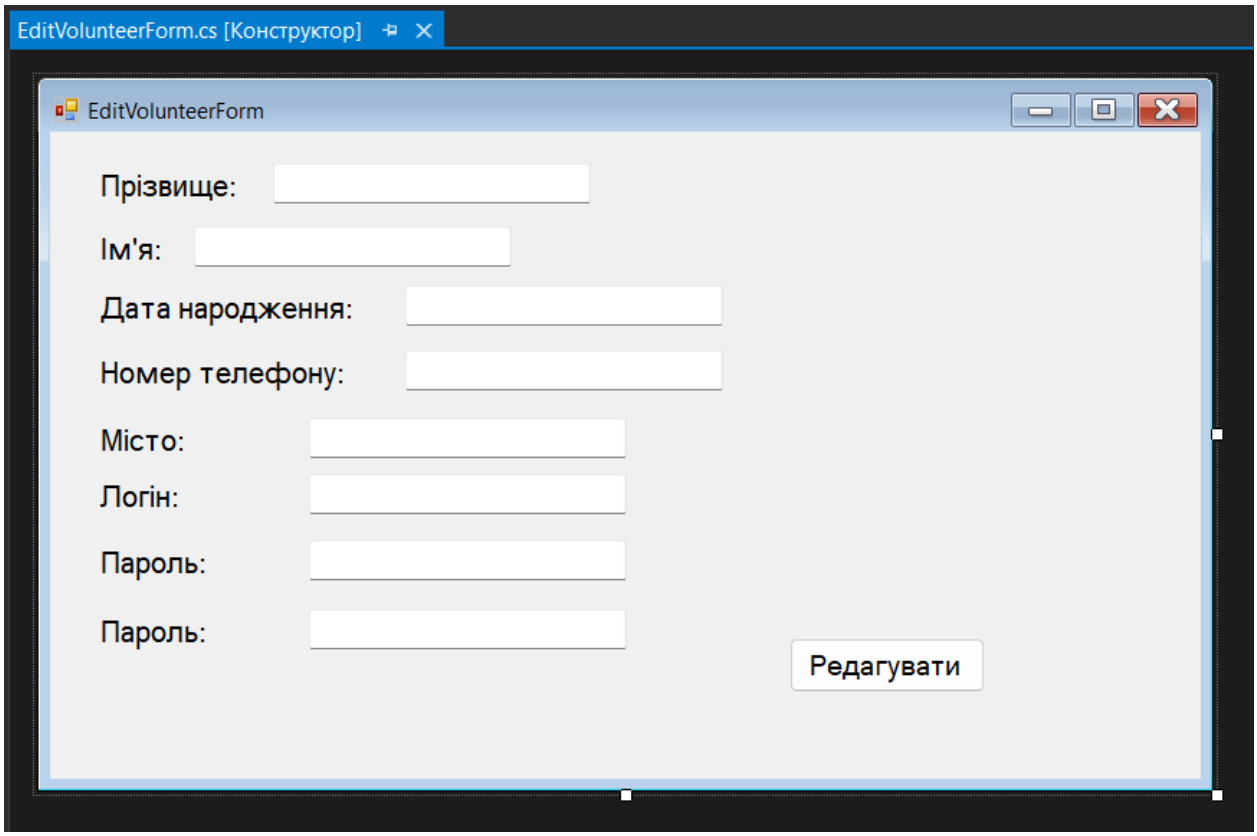


Рисунок 3.18 – EditVolunteerForm.cs

Створити форму Form1.cs (рис. 3.19), яка забезпечує можливість пошуку замовників в системі(прізвище, ім’я, телефон), додавання нових замовників та інформацію про волонтера.

Зокрема:

- ButtonClearLabel_Click(object sender, EventArgs e) – на вхід приймає

параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – очищує дані, які були введені для пошуку.

- ButtonSearchLabel_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – знаходження замовника по прізвищу, імені та/або номеру телефону.
- linkLabelVolunteer_LinkClicked(object sender, LinkLabelLinkClicked EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відкриває інформацію про волонтера.
- ButtonAddCustomer_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додає нового замовника в систему.

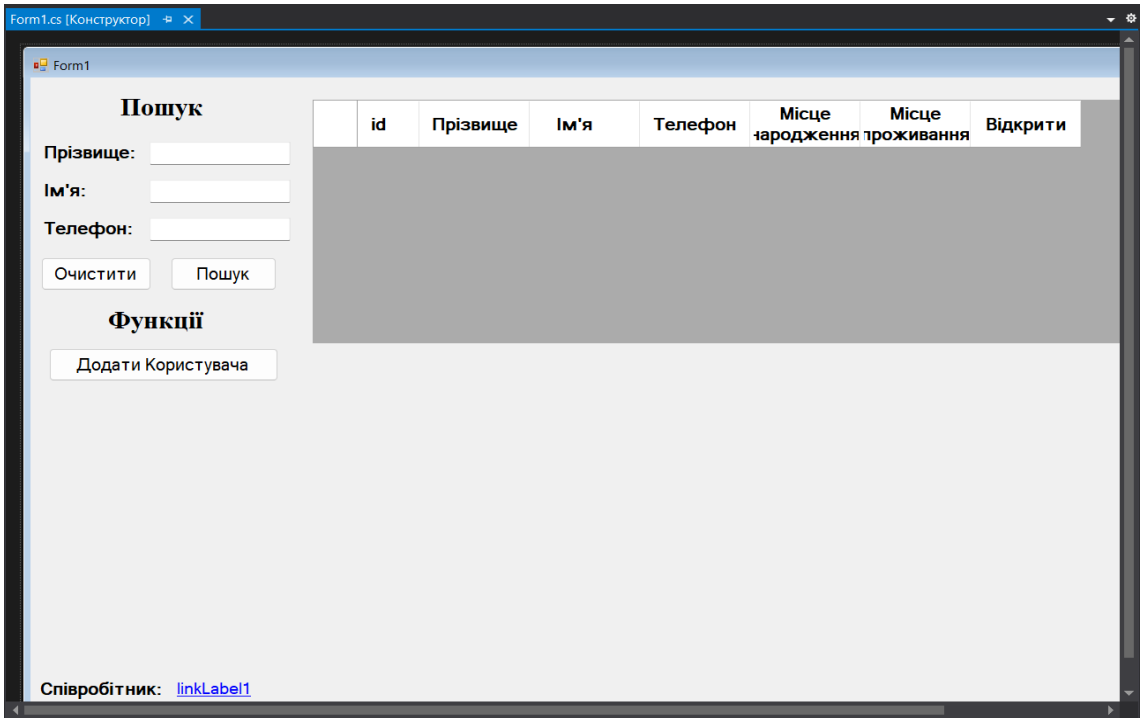


Рисунок 3.19 – Form1.cs

Створити форму LoginForm.cs (рис. 3.20), вона потрібна для авторизації волонтерів в систему.

Зокрема:

- buttonOk_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – вхід в систему під унікальним логіном та паролем .
- buttonClear_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – очищує поля, які були введені раніше.

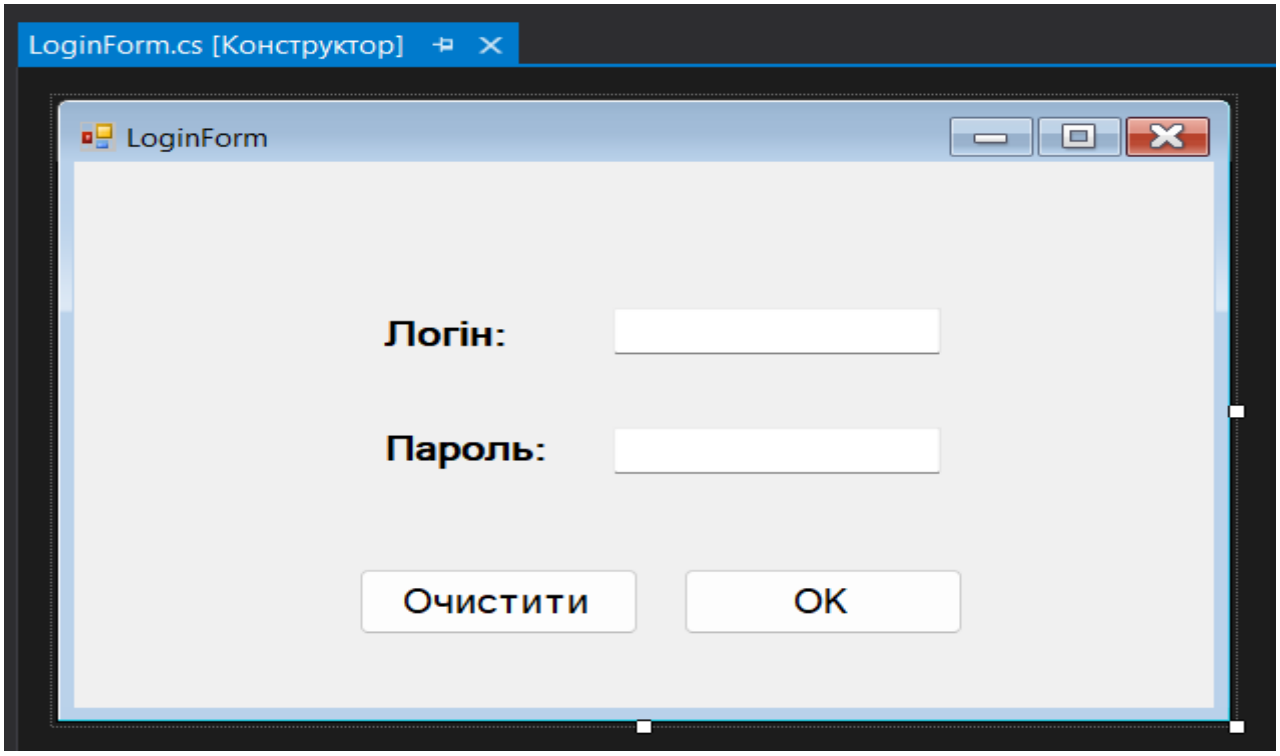


Рисунок 3.20 – LoginForm.cs

Створити форму MovePackageForm.cs (рис. 3.21), вона потрібна для додавання вже існуючих пакунків в систему.

Зокрема:

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який

містить посилання на елемент керування/об'єкт, що спричинив подію – оновлення даних щодо кількості пакунку, який вже був в системі.

- IsDigitsOnly(string str) – на вхід приймає рядок – перевіряє чи є заданий рядок числом.

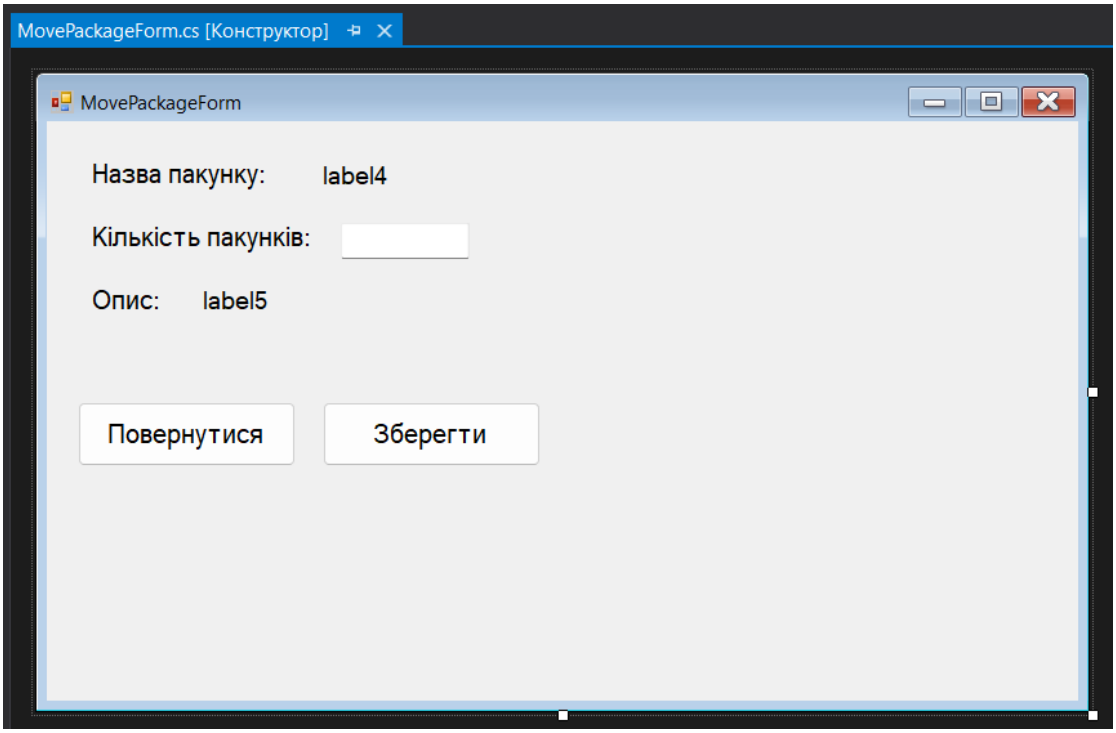


Рисунок 3.21 – MovePackageForm.cs

Створити форму ShowAllDelivery.cs (рис. 3.22), вона потрібна для відображення пакунків, які є в наявності та їх кількість.

Зокрема:

- AllPackageLoad() – на вхід не приймає значень – відображення інформацію про пакунки в системі.

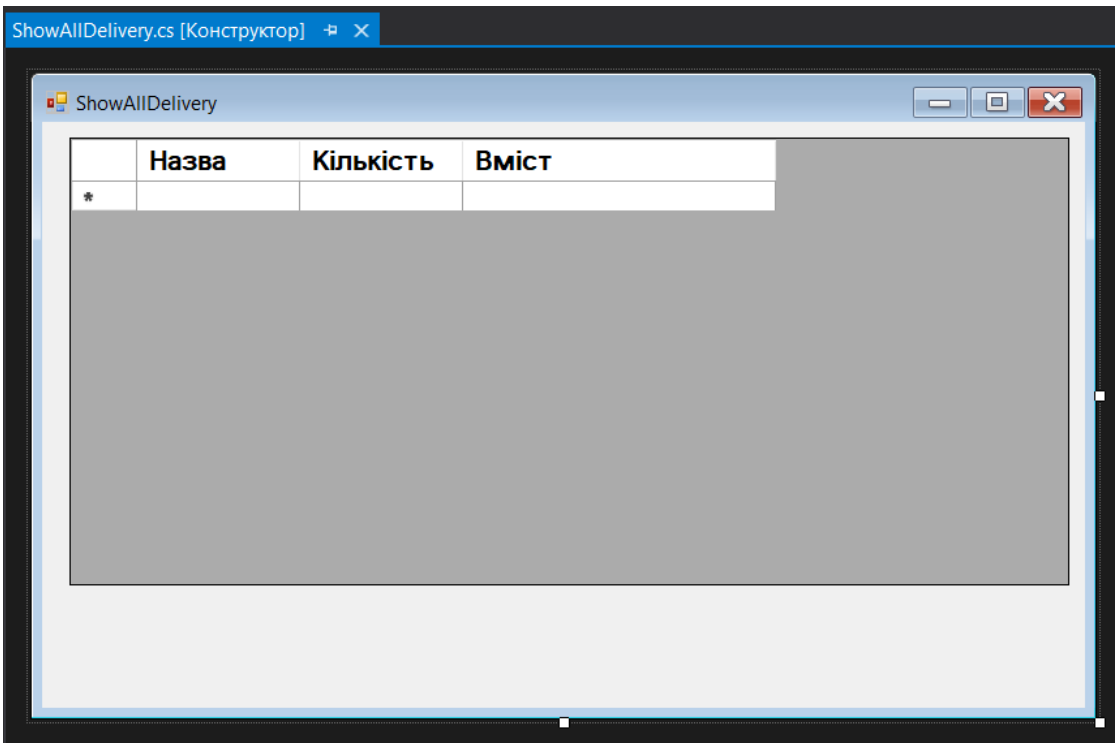


Рисунок 3.22 – ShowAllDelivery.cs

Створити форму ShowCustomCardForm.cs (рис. 3.23), вона потрібна для відображення інформації про конкретного замовника, про видачі замовнику та можливість редагувати інформацію про замовника та видачі.

Зокрема:

- ShowDataGridDelivery() – на вхід не приймає значень – відображення інформації про видачі у замовника.
- ShowCustomerInfo()– на вхід не приймає значень – відображення інформації про замовника.
- AddDeliveryButton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – додавання видачі замовнику.
- DeleteCustomerButton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – видалення замовника із системи.
- EditCustomerButton_Click(object sender, EventArgs e) – на вхід

приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – редагування інформації про замовника.

- ShowAllDeliveryButton_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – відображення інформації про всі видачі.

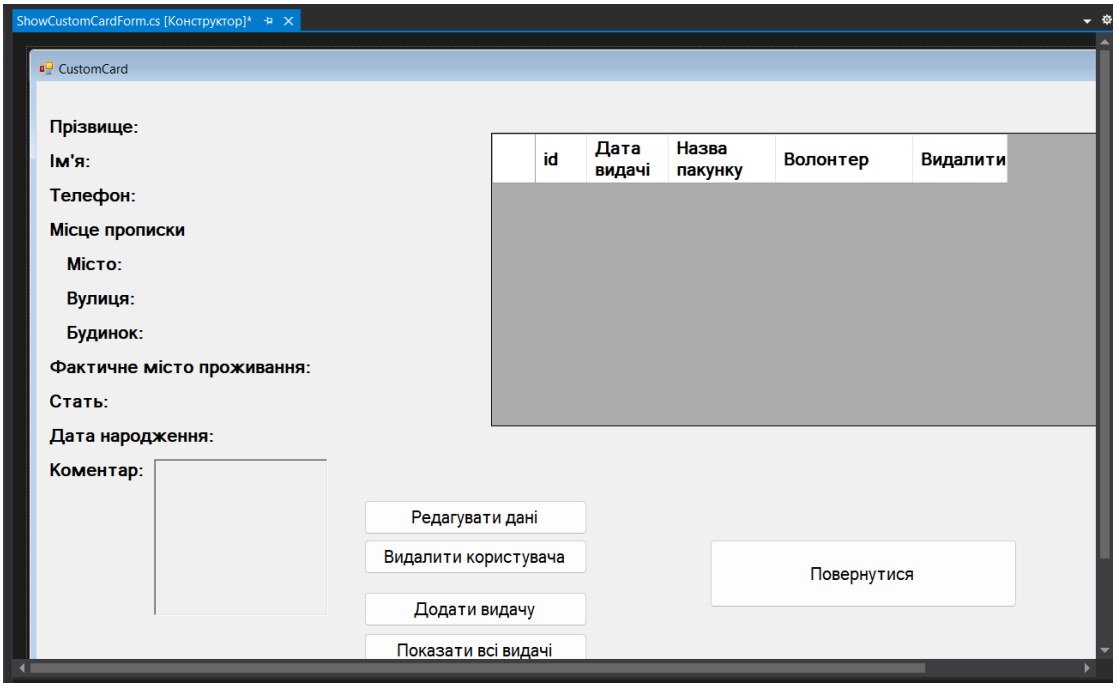


Рисунок 3.23 – ShowCustomCardForm.cs

Створити форму VolunteerCardForm.cs (рис. 3.24), вона потрібна для відображення інформації про конкретного волонтера в системі.

Зокрема:

- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – повернутися в попереднє вікно.

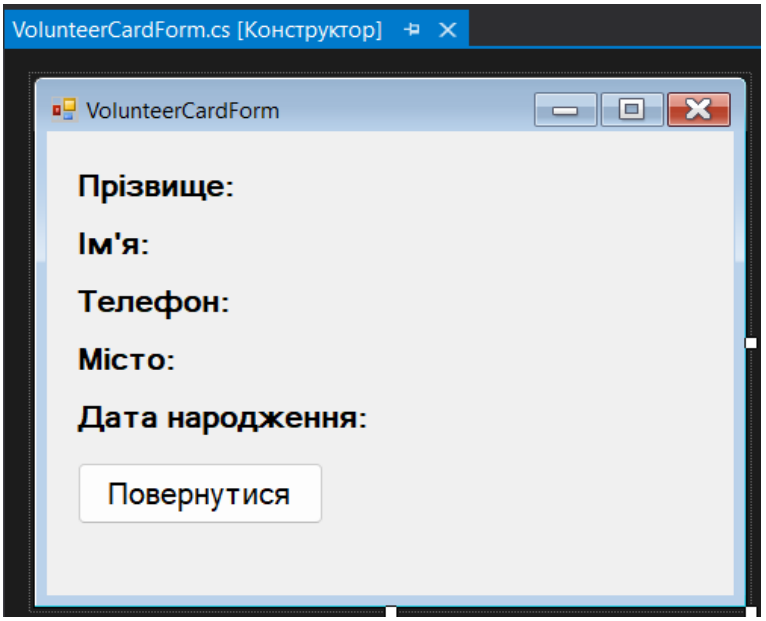


Рисунок 3.24 – VolunteerCardForm.cs

Створити форму VolunteerListForm.cs (рис. 3.25), вона потрібна для відображення інформації про всіх волонтерів в системі. Також має бути можливість додавати нового волонтера в систему, та формувати по існуючим волонтерам звіт.

Зокрема:

- buttonAddVolunteer_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – повернутися в попереднє вікно.
- button1_Click(object sender, EventArgs e) – на вхід приймає параметр e, який містить дані про подію та параметр під назвою sender, який містить посилання на елемент керування/об'єкт, що спричинив подію – формування звіту MS Word.

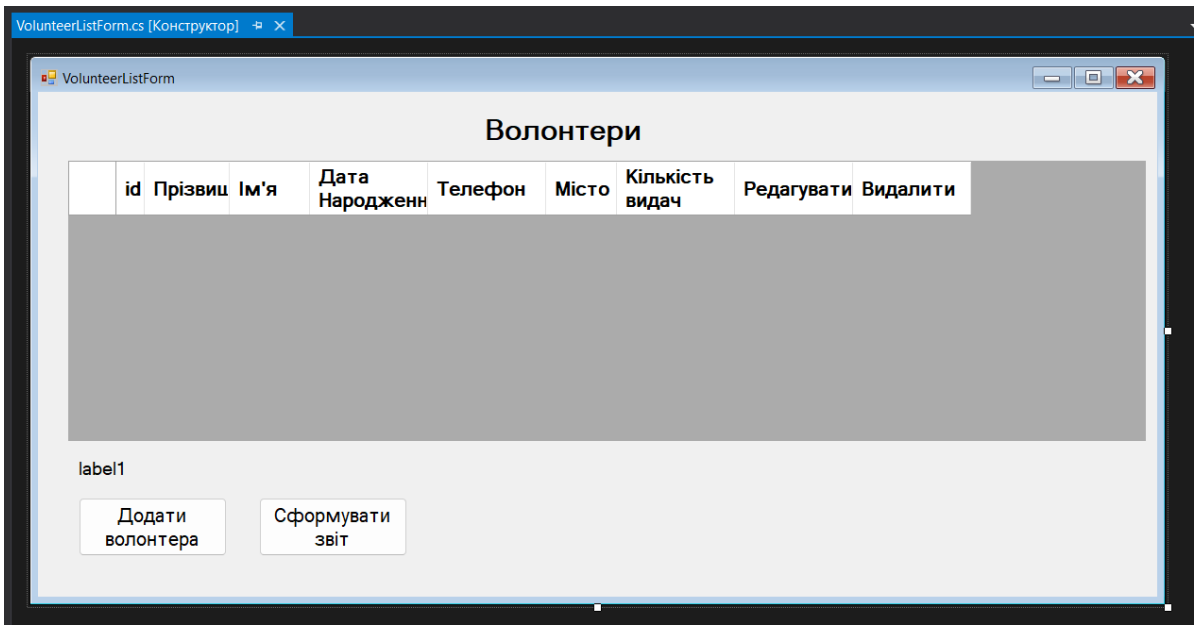


Рисунок 3.25 – VolunteerListForm.cs

Повний лістинг коду модулів даної розробки наведено в додатку В.

3.4 Використання програмного додатку

Розпочинаємо роботу додатком з вікна авторизації (рис. 3.26). Потрібно ввести правильний логін та пароль.

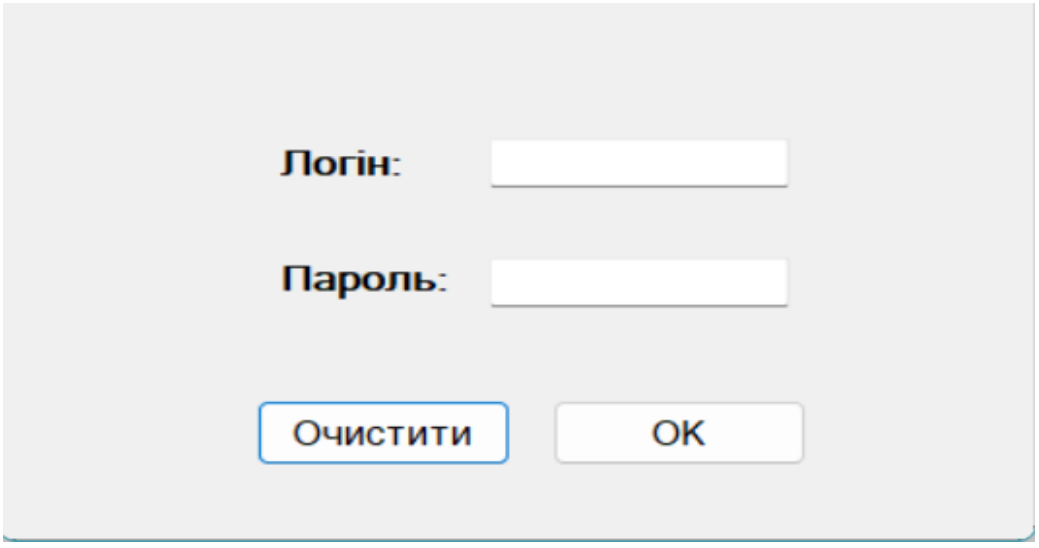


Рисунок 3.26 – Вікно авторизації

Після введення логіну та паролю, натискаємо кнопку ОК (рис. 3.27).

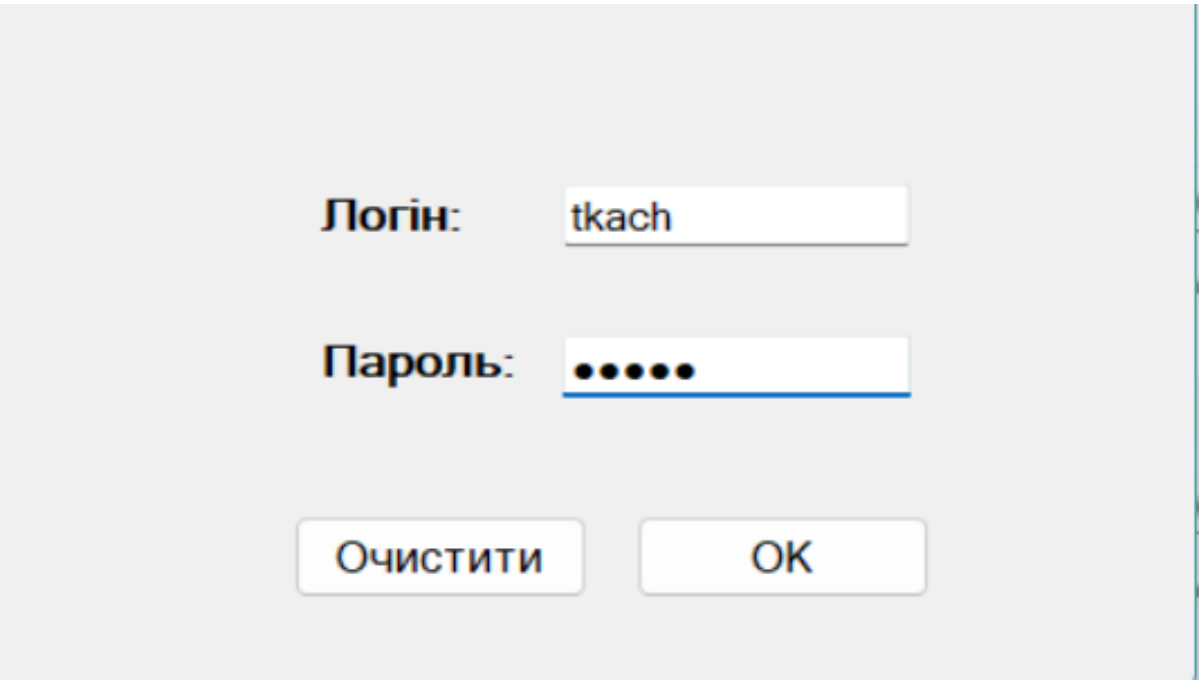


Рисунок 3.27 – Вікно авторизації після введення даних для входу

Якщо дані для входу введені вірно, то потрапляємо на головну сторінку волонтера (рис. 3.28). Можемо знайти замовника за прізвищем, ім’ям та/або телефоном. Знаходимо замовників в системі за ім’ям Павло (рис. 3.29). Можемо додати нового замовника в систему через кнопку «Додати Користувача» (рис. 3.30). Також можна переглянути інформацію про волонтера (рис. 3.31).

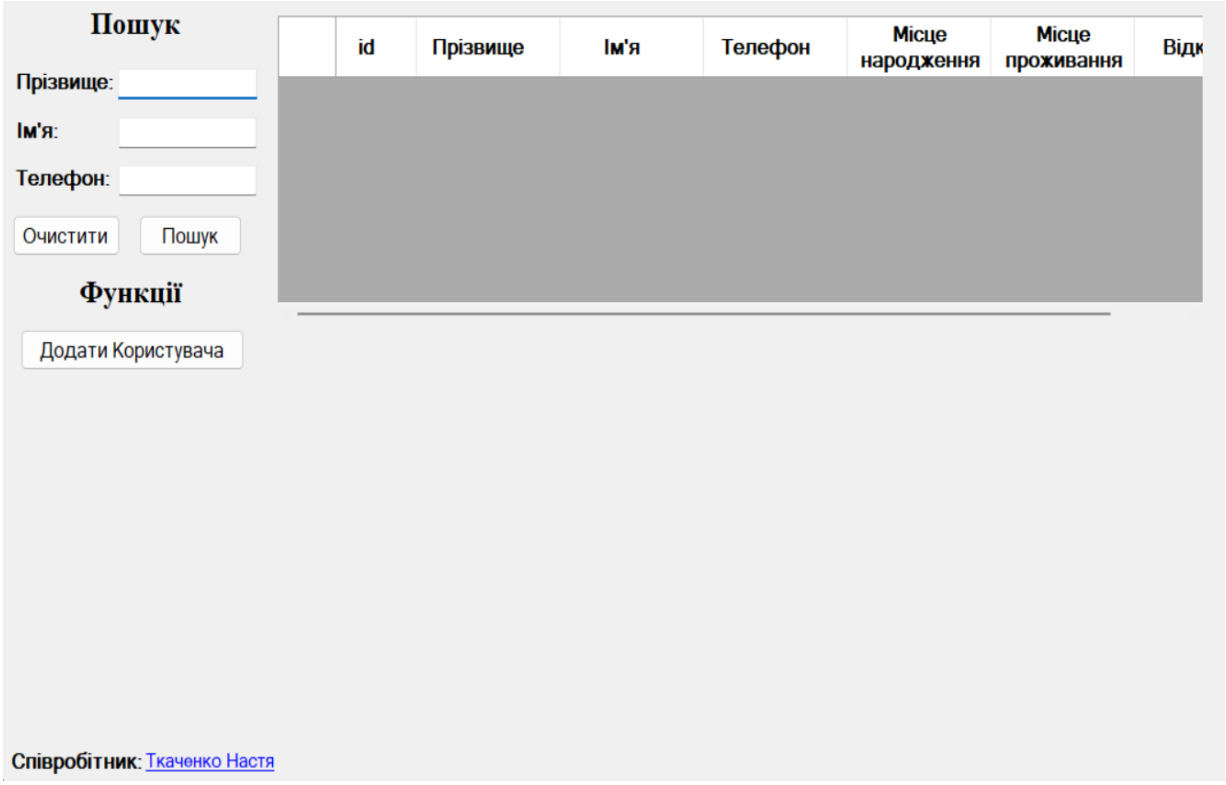


Рисунок 3.28 – Головна сторінка волонтера

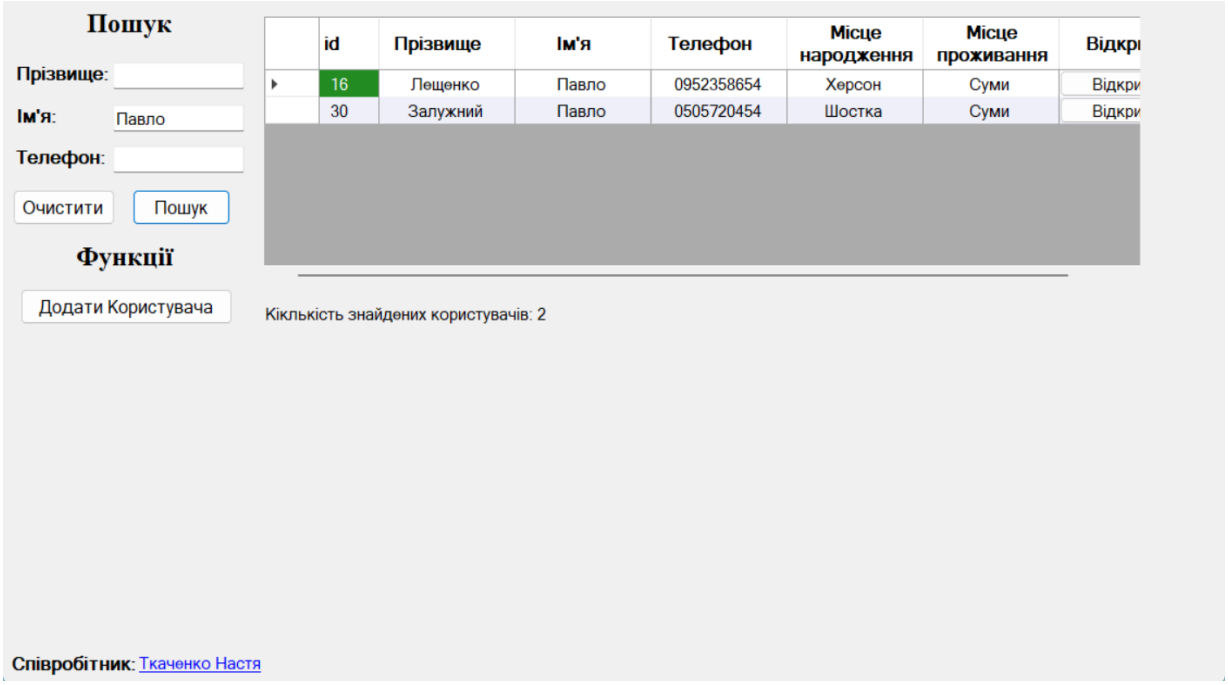


Рисунок 3.29 – Пошук замовника за іменем «Павло»

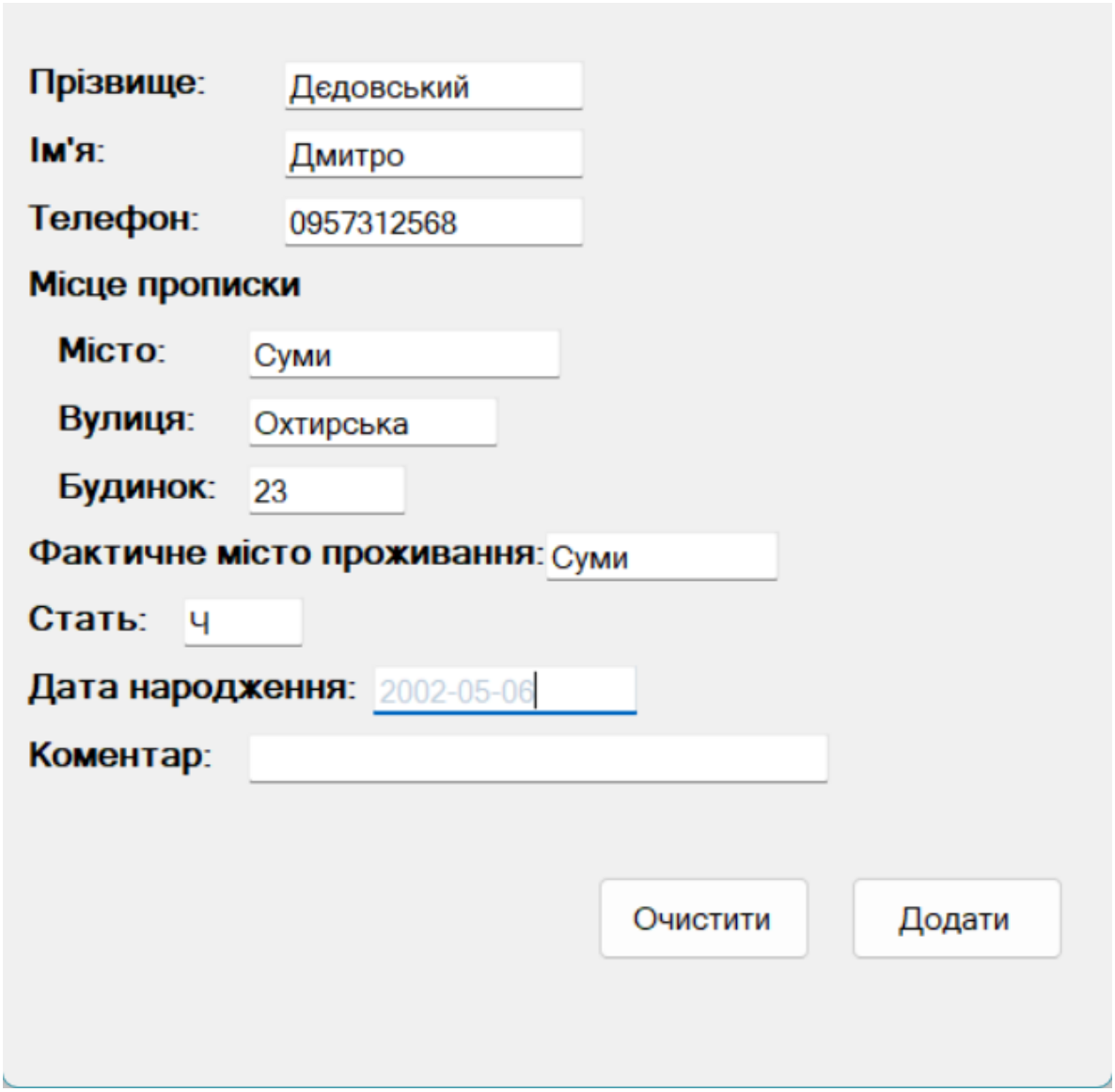


Рисунок 3.30 – Вікно додавання нового замовника в систему

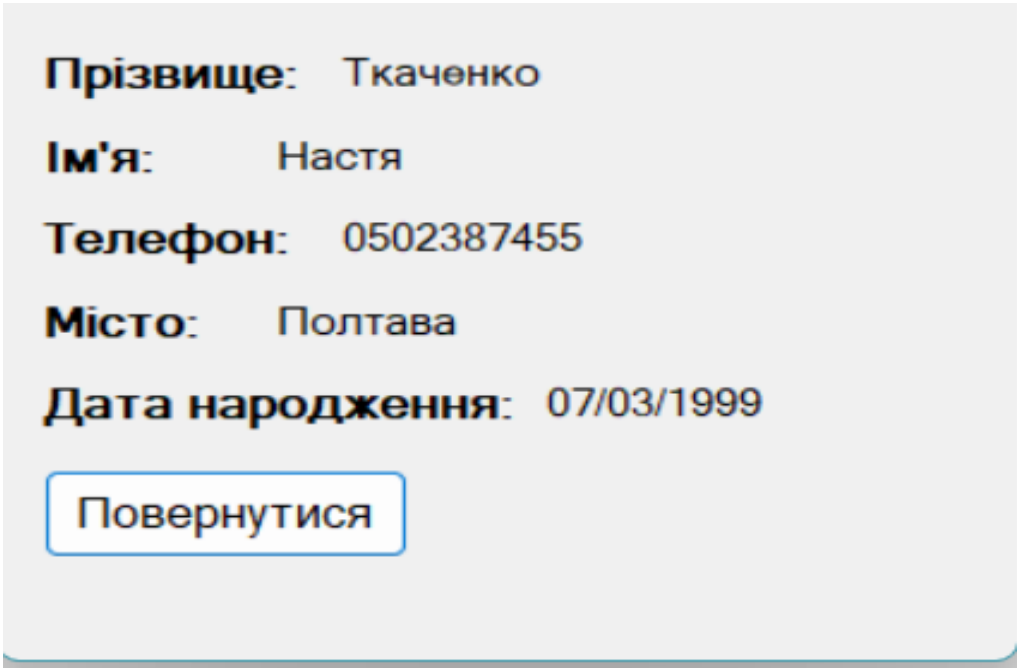


Рисунок 3.31 – Відображення інформації про волонтера

Відкриваємо вікно карти замовника в системі (рис. 3.32). Натискаючи кнопку «Редагувати дані» відкриваємо вікно редагування даних про замовника (рис. 3.33). Натискаючи кнопку «Видалити користувача» з’являється вікно підтвердження наших дій (рис. 3.34). Натискаючи кнопку «Додати видачу» з’являється вікно додавання видачі цьому замовнику (рис. 3.35). Натискаючи кнопку «Показати всі видачі» з’являється вікно з інформацією про видачі, які є наявні на даний момент (рис. 3.36).

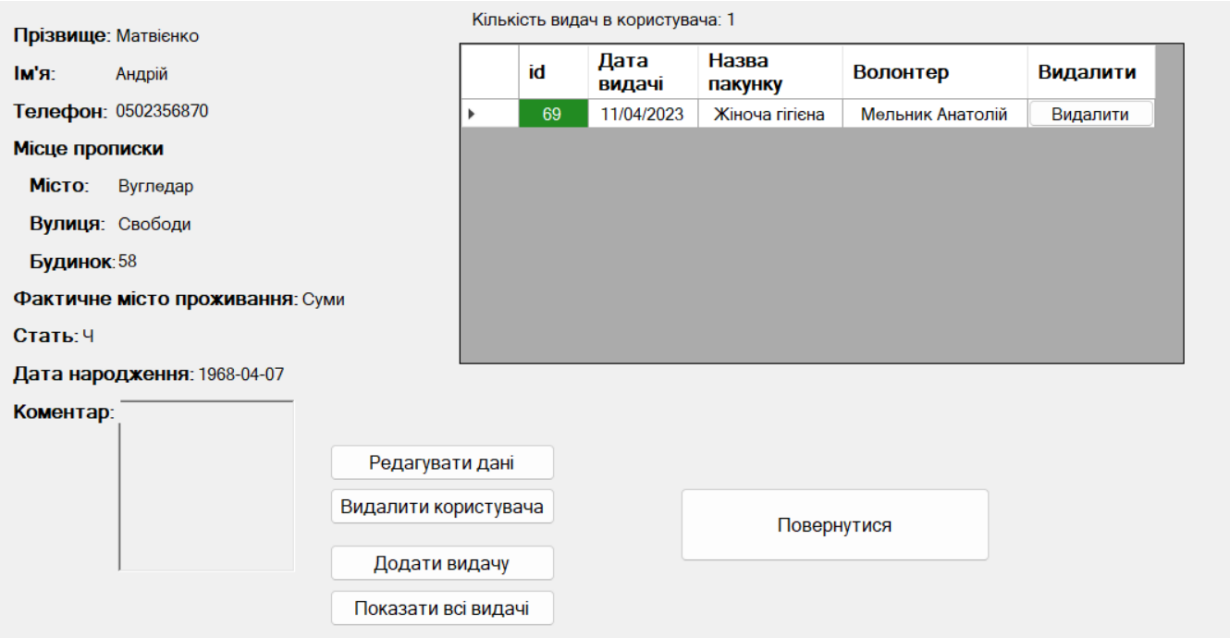


Рисунок 3.32 – Вікно карти замовника

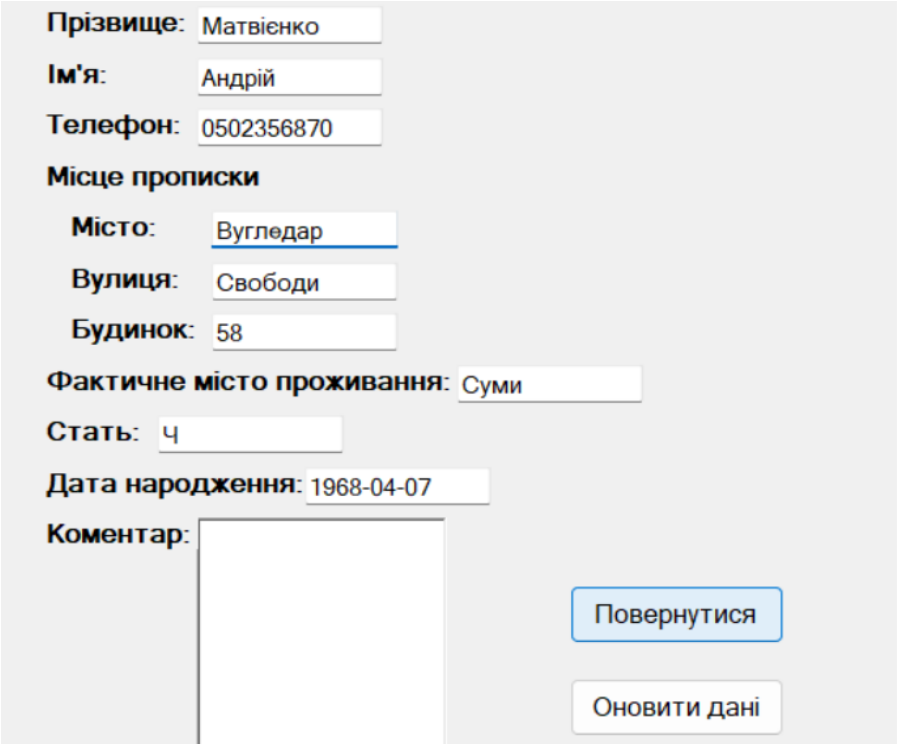


Рисунок 3.33 – Вікно редагування даних про замовника

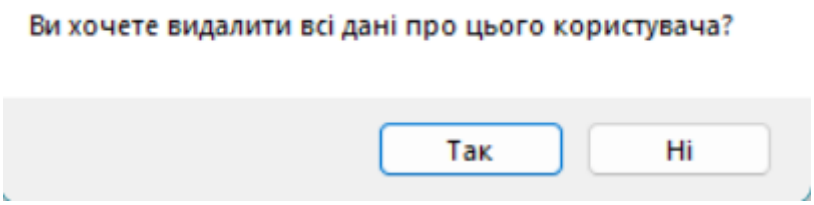


Рисунок 3.34 – Вікно видалення всіх даних про замовника

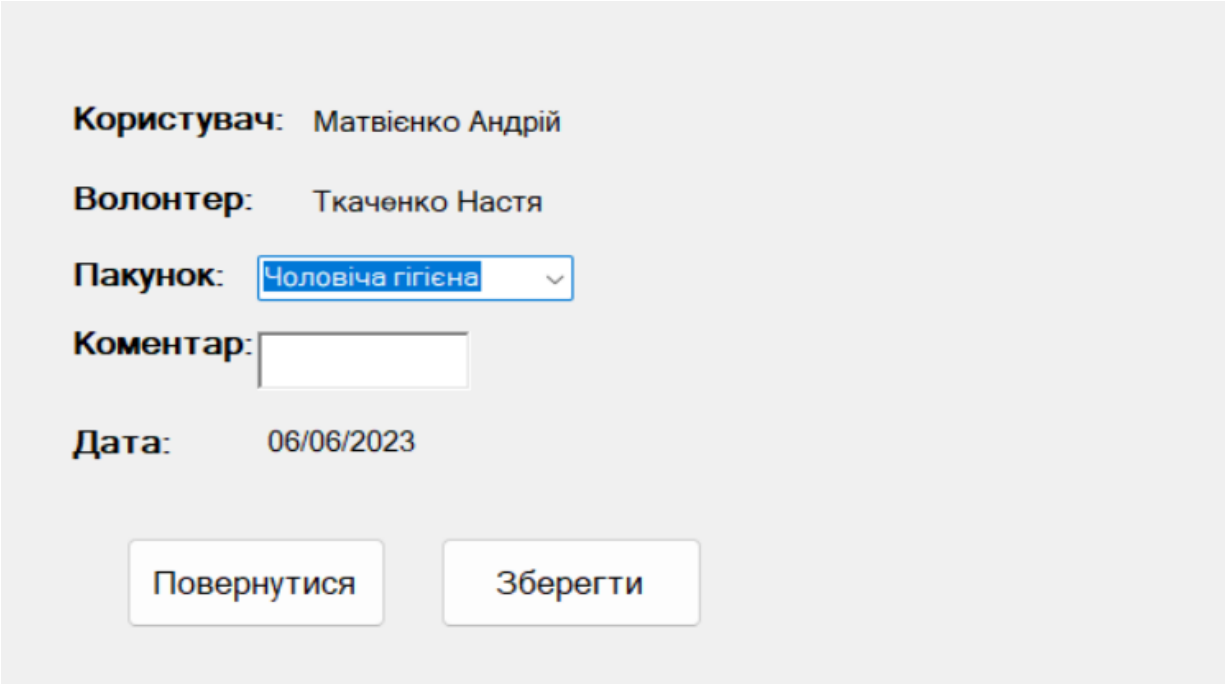


Рисунок 3.35 – Вікно додавання видачі конкретному замовнику

Рисунок 3.36 – Вікно відображення всіх наявних пакунків

Панель адміністратора має наступний вигляд (рис. 3.37). Натискаючи кнопку «Волонтери» з’являється вікно про волонтерів в системі (рис. 3.38). Натискаючи кнопку «Пакунки» з’являється вікно про пакунки в системі (рис. 3.40). Натискаючи кнопку «Статистика» з’являється вікно з статистикою видачі пакунків (рис. 3.42).

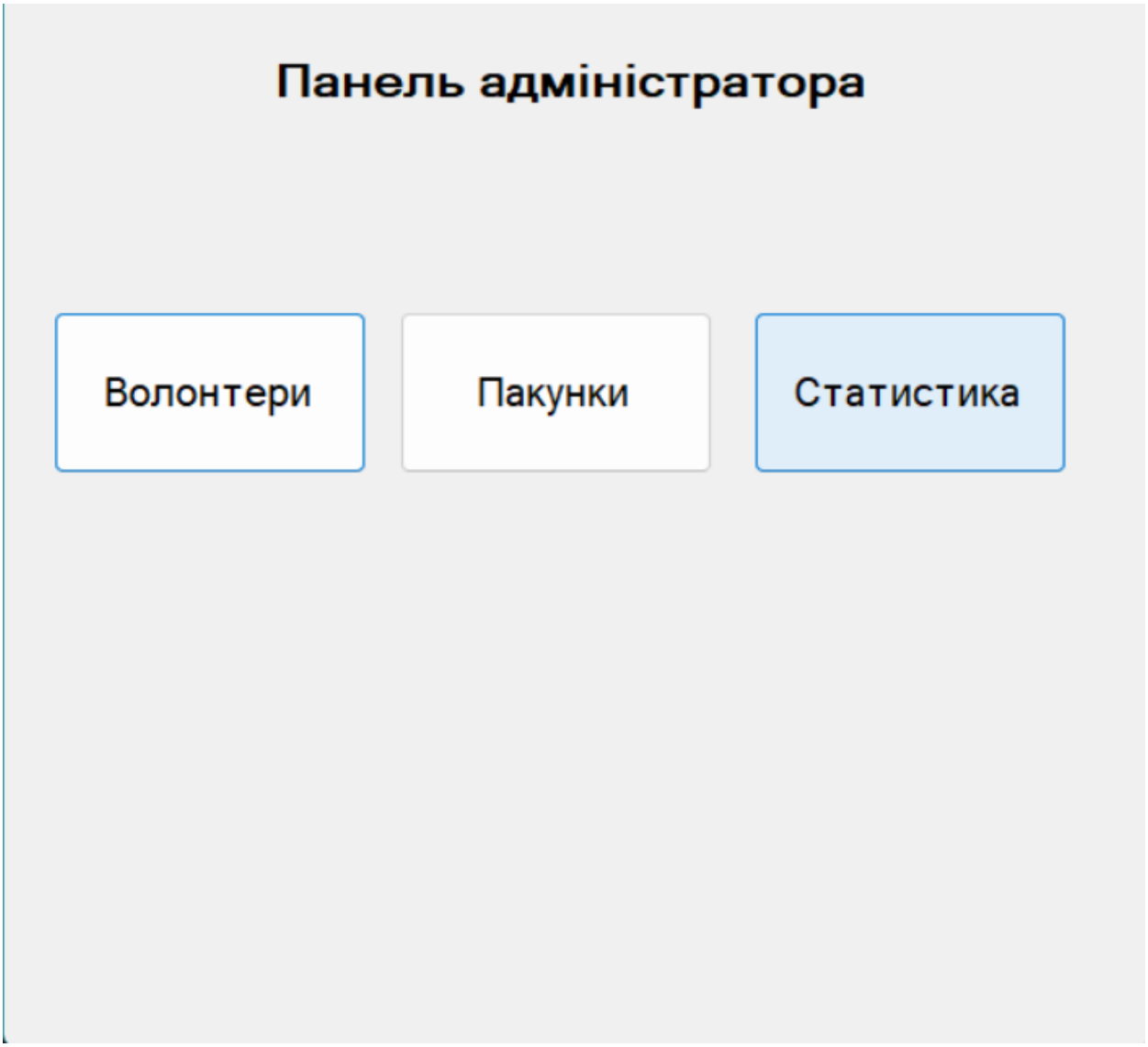


Рисунок 3.37 – Панель адміністратора

Вікно інформації про волонтерів відображає інформацію про волонтерів в системі (рис. 3.38). Є можливість редагувати дані про волонтерів та/або видаляти конкретного волонтера. Є можливість додати нового волонтера в систему та сформувати звіт (рис. 3.39).

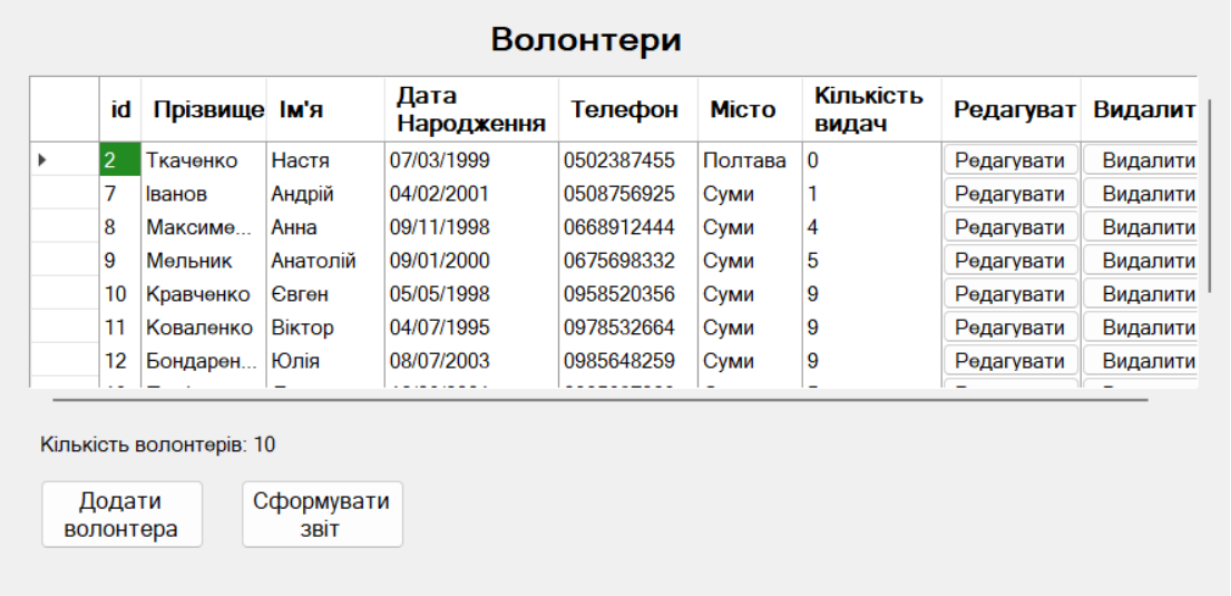


Рисунок 3.38 – Вікно інформації про волонтерів

Сформований звіт по волонтерам має наступний вигляд (рис. 3.39).

Кількість Волонтерів 10
Список Волонтерів:

id	Прізвище	Ім'я	Дата Народження	Телефон	Місто
2	Ткаченко	Настя	07/03/1999	0502387455	Полтава
7	Іванов	Андрій	04/02/2001	0508756925	Суми
8	Максименко	Анна	09/11/1998	0668912444	Суми
9	Мельник	Анатолій	09/01/2000	0675698332	Суми
10	Кравченко	Євген	05/05/1998	0958520356	Суми
11	Коваленко	Віктор	04/07/1995	0978532664	Суми
12	Бондаренко	Юлія	08/07/2003	0985648259	Суми
13	Поліщук	Дмитро	12/03/2001	0995687023	Суми
14	Гончаров	Олексій	10/11/2001	0735670230	Суми
15	Прокопенко	Світлана	10/04/1999	0682055881	Суми

Рисунок 3.39 – MS Word звіт по волонтерам

Вікно інформації про пакунки відображає інформацію про пакунки в системі (рис. 3.40). Є можливість редагувати дані про пакунки. Є можливість додати новий пакунок в систему та сформувати звіт (рис. 3.41).

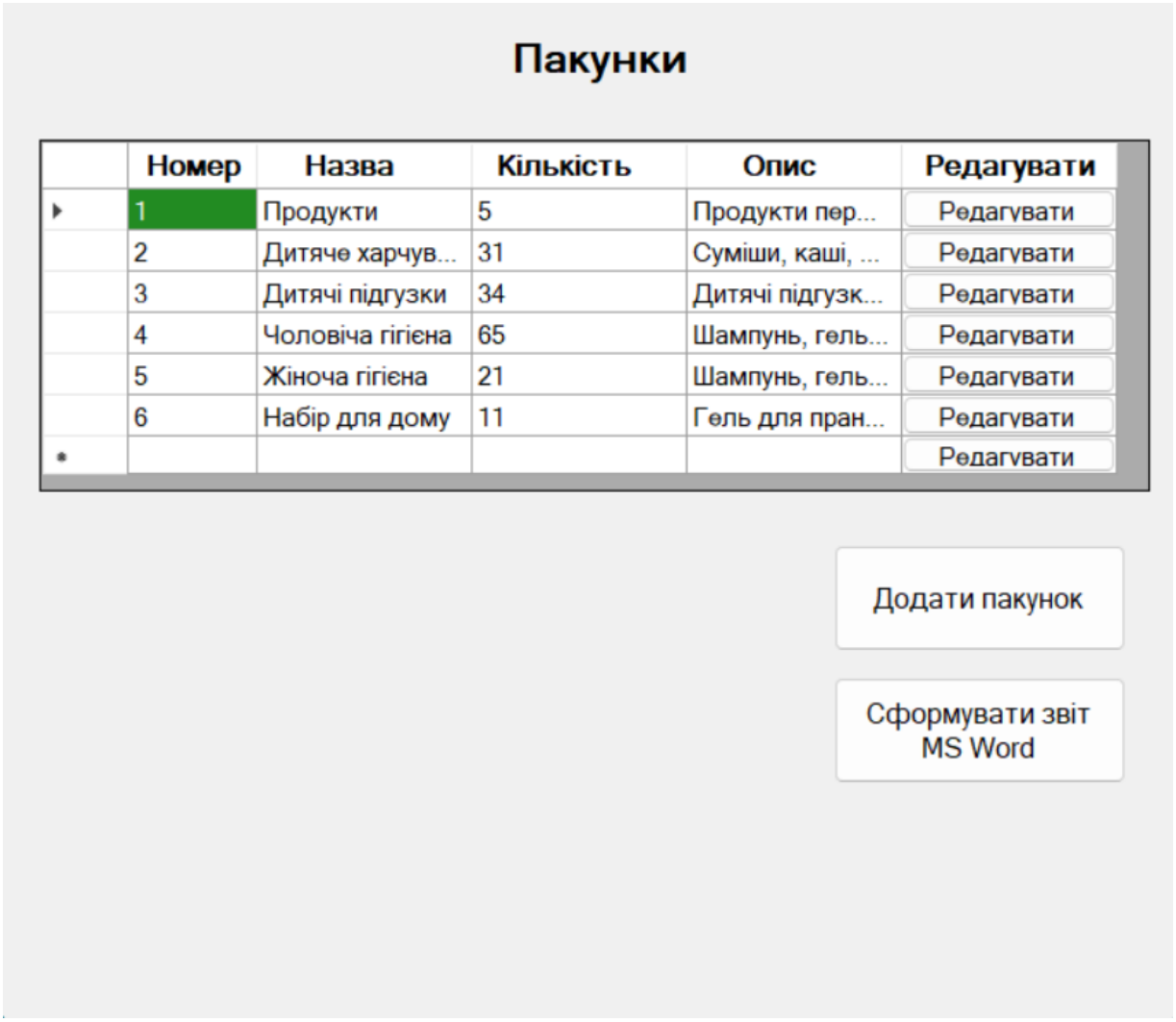


Рисунок 3.40 – Вікно інформації про пакунки

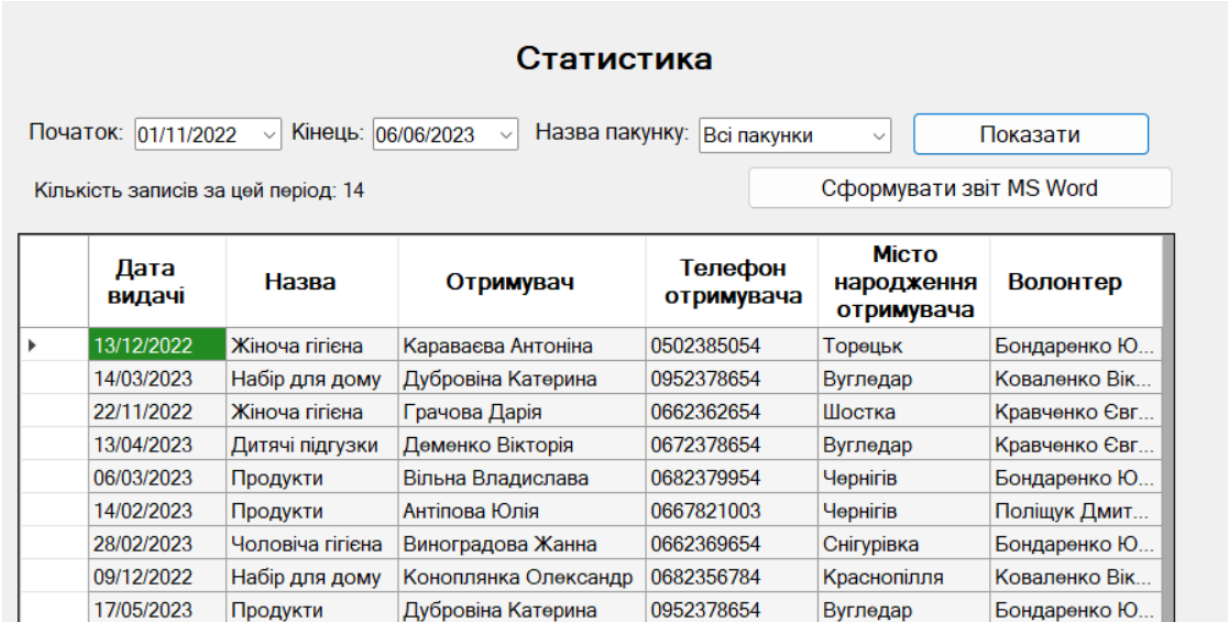
Сформований звіт по пакункам має наступний вигляд (рис. 3.41)

Список Пакунків:

Номер	Назва	Кількість	
1	Продукти	5	Продукти першої необхідності
2	Дитяче харчування	31	Суміши, каші, пюре
3	Дитячі підгузки	34	Дитячі підгузки різних розмірів
4	Чоловіча гігієна	65	Шампунь, гель, піна для гоління, антиперспірант
5	Жіноча гігієна	21	Шампунь, гель, зубна паста, крем, прокладки
6	Набір для дому	11	Гель для прання, засіб для миття посуду, засіб для миття підлоги

Рисунок 3.41 – MS Word звіт по пакункам

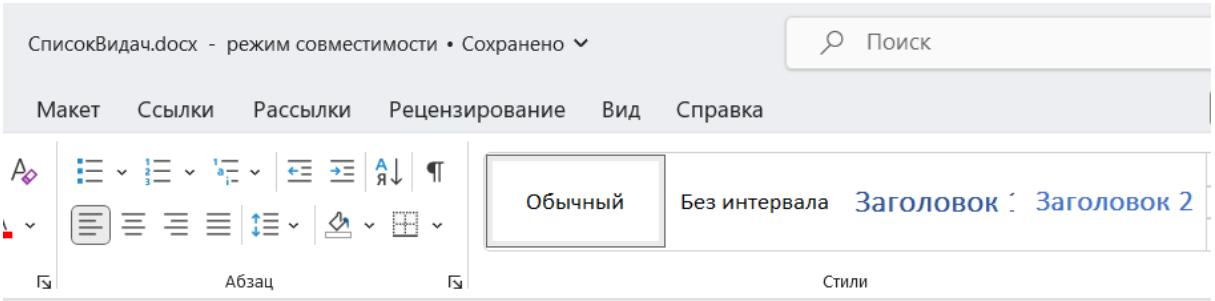
Вікно статистики відображає інформацію про видані пакунки за певний період часу з можливістю вибору статистики по одному пакунку (рис. 3.42). Є можливість сформувати звіт по пакункам (рис. 3.43).



Дата видачі	Назва	Отримувач	Телефон отримувача	Місто народження отримувача	Волонтер
13/12/2022	Жіноча гігієна	Караваєва Антоніна	0502385054	Торецьк	Бондаренко Ю...
14/03/2023	Набір для дому	Дубровіна Катерина	0952378654	Вугледар	Коваленко Вік...
22/11/2022	Жіноча гігієна	Грачова Дарія	0662362654	Шостка	Кравченко Євг...
13/04/2023	Дитячі підгузки	Деменко Вікторія	0672378654	Вугледар	Кравченко Євг...
06/03/2023	Продукти	Вільна Владислава	0682379954	Чернігів	Бондаренко Ю...
14/02/2023	Продукти	Антіпова Юлія	0667821003	Чернігів	Поліщук Дмит...
28/02/2023	Чоловіча гігієна	Виноградова Жанна	0662369654	Снігурівка	Бондаренко Ю...
09/12/2022	Набір для дому	Коноплянка Олександр	0682356784	Краснопілля	Коваленко Вік...
17/05/2023	Продукти	Дубровіна Катерина	0952378654	Вугледар	Бондаренко Ю...

Рисунок 3.42 – Вікно інформації про видані пакунки

Сформований звіт має наступний вигляд (рис. 3.43).



Кількість Видач за період з 01/11/2022 по 06/06/2023: 14

Список Видач за період з 01/11/2022 по 06/06/2023:

Дата видачі	Назва	Отримувач	Телефон отримувача	Місто народження отримувача	Волонтер
13/12/2022	Жіноча гігієна	Караваєва Антоніна	0502385054	Торецьк	Бондаренко Юлія
14/03/2023	Набір для дому	Дубровіна Катерина	0952378654	Вугледар	Коваленко Віктор
22/11/2022	Жіноча гігієна	Грачова Дарія	0662362654	Шостка	Кравченко Євген
13/04/2023	Дитячі підгузки	Деменко Вікторія	0672378654	Вугледар	Кравченко Євген
06/03/2023	Продукти	Вільна Владислава	0682379954	Чернігів	Бондаренко Юлія

Рисунок 3.43 – MS Word звіт по виданим пакункам

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи бакалавра було створено програмний додаток підтримки діяльності волонтерської організації.

На початку реалізації проекту було проаналізовано предметну область. Визначено актуальність роботи. Під час аналізу програмних продуктів-аналогів були визначені їх сильні та слабкі сторони. Зроблено висновки, які дозволять уникнути повторення їх помилок та використати найбільш ефективні функції, що були доступні. Новий програмний продукт буде відрізнятися від аналогів цільовим функціоналом, який спраямований на забезпеченя реалізації конкретних функцій волонтерських організацій, які займаються видачею пакунків допомоги, що зробить його незамінним для волонтерської організації. Отримані результати після порівняння аналогів дозволяють стверджувати, що створення такого продукту на сьогоднішній день є актуальним та доцільним.

Було сформовано мету та задачі проекту. Був проведений аналіз засобів реалізації та обрано ті інструменти, які будуть використані. Сформовано технічне завдання на розробку даного проекту та виконано планування робіт по реалізації проекту.

Виконано моделювання принципу роботи додатку по забезпеченню основної функції – підтримка діяльності влонтерів (модель в нотації IDEF0). Розроблено графічне представлення роботи програмного додатку в нотації UML. Створено фізичну модель даних бази дани.

Розроблена архітектура програмного додатку з використанням HLD діаграми. Представлена програмна реалізація головних елементів програмного додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волонтерські організації, які зараз працюють в Україні [Електронний ресурс] – Доступ до ресурсу: https://finance.ua/ua/saving/volonterskie-organizacii-v-ukraine (Дата звернення: 20.05.23)

2. Ukraine Refugee Situation [Електронний ресурс] – Доступ до ресурсу: https://data2.unhcr.org/en/situations/ukraine (Дата звернення: 20.05.23)

3. Russia’s war of aggression against Ukraine generates historic migration flows: More support needed for integration now and possible future return [Електронний ресурс] – Доступ до ресурсу: https://www.oecd.org/newsroom/russia-s-war-of-aggression-against-ukraine-generates-historic-migration-flows.htm (Дата звернення: 20.05.23)

4. Дєдовський Д.І., Ващенко С.М. Програмний засіб підтримки діяльності волонтерського центру. // «Інформатика, математика, автоматика»: матеріали та програма науково-технічної конференції, м. Суми, 27 – 28 квітня 2023 р. – Суми: Сумський державний університет, 2023.

5. What is a Desktop Application? [Електронний ресурс] – Доступ до ресурсу: https://whatfix.com/blog/desktop-application/ (Дата звернення: 20.05.23)

6. Web, Mobile or Desktop App – Which is Right for Your Project? [Електронний ресурс] – Доступ до ресурсу: https://arrowcore.com/blogs/web-mobile-or-desktop-app-which-is-right-for-your-project/ (Дата звернення: 20.05.23)

7. Desktop додатки [Електронний ресурс] – Доступ до ресурсу: https://wezom.com.ua/ua/blog/desktop-dodatok (Дата звернення: 20.05.23)

8. Цілі, завдання та функції волонтерської діяльності в суспільстві [Електронний ресурс] – Доступ до ресурсу: https://studfile.net/preview/9213462/page:2/ (Дата звернення: 20.05.23)

9. Best Volunteer Management Software [Електронний ресурс] – Доступ до ресурсу: https://www.g2.com/categories/volunteer-management (Дата звернення: 20.05.23)

10. Digitalization in daily life [Електронний ресурс] – Доступ до ресурсу: https://www.desightstudio.com/en/digitalization-in-daily-life/ (Дата звернення: 20.05.23)

11. Структура додатка Windows [Електронний ресурс] – Доступ до ресурсу: https://studfile.net/preview/9864558/page:16/ (Дата звернення: 20.05.23)

12. Огляд можливостей Excel [Електронний ресурс] – Доступ до ресурсу: http://nikolay.in.ua/navchaemos/ms-excel/znajomstvo-z-excel-2010/290-oglyad-mozhlivostej-excel (Дата звернення: 20.05.23)

13. Огляд можливостей Dilovod роздрібної торгівлі [Електронний ресурс] – Доступ до ресурсу: https://kubik.com.ua/ua/articles/dilovod-trade (Дата звернення: 20.05.23)

14. Розробка додатків засобами мови програмування C# [Електронний ресурс] – Доступ до ресурсу: https://www.researchgate.net/publication/354860614_Rozrobka_dodatkiv_zasobami_movi_programuvanna_C (Дата звернення: 20.05.23)

15. База даних MySQL [Електронний ресурс] – Доступ до ресурсу: https://promoter.net.ua/articles/baza-danix-mysql.html (Дата звернення: 20.05.23)

16. Складові системи планування робіт [Електронний ресурс] – Доступ до ресурсу: https://buklib.net/books/23851/ (Дата звернення: 20.05.23)

17. Постановка цілей по SMART – приклади, критерії [Електронний ресурс] – Доступ до ресурсу: https://pdatu.edu.ua/images/vihovna-robota/psiholog/ps10.pdf (Дата звернення: 20.05.23)

18. WBS структура проекту [Електронний ресурс] – Доступ до ресурсу: https://studfile.net/preview/5680685/page:12/ (Дата звернення: 20.05.23)

19. What are Gantt charts? [Електронний ресурс] – Доступ до ресурсу: https://www.atlassian.com/agile/project-management/gantt-chart (Дата звернення: 20.05.23)

20. Ризики проекту [Електронний ресурс] – Доступ до ресурсу: https://skillsetter.io/blog/risk-management-ua (Дата звернення: 20.05.23)

21. Онлайн-бухгалтерія Dilovod [Електронний ресурс] – Доступ до ресурсу: https://dilovod.ua/ (Дата звернення: 20.05.23)

22. Функціональна схема IDEF0 [Електронний ресурс] – Доступ до ресурсу: http://um.co.ua/13/13-1/13-121860.html (Дата звернення: 20.05.23)

23. Use-case diagrams [Електронний ресурс] – Доступ до ресурсу: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (Дата звернення: 20.05.23)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інформаційної системи
«Програмний додаток підтримки діяльності волонтерського центру»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

Ващенко С.М.

Студент групи ІТ-91

Дєдовський Д.І.

Суми 2023

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ ДЕСКТОП-ДОДАТКУ

1.1 Призначення десктоп-додатку

Додаток дозволить співробітникам(волонтерам) переглядати, додавати, видаляти інформацію про отримувачів допомоги. Також десктоп-додаток створить зручність ведення та аналізу звітності для адміністратора.

1.2 Мета створення десктоп-додатку

Головна мета створення додатку – автоматизація управління процесами у волонтерській організації та збільшення продуктивності роботи волонтерів.

1.3 Цільова аудиторія

До цільової аудиторії десктоп-додатку відносяться співробітники (волонтери) волонтерської організації та адміністратор.

2 ВИМОГИ ДО ДЕСКТОП-ДОДАТКУ

2.1 Вимоги до десктоп-додатку в цілому

2.1.1 Вимоги до структури й функціонування десктоп-додатку

Додаток має забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути представлений у вигляді десктоп-додатку, який містить якісне інформаційне наповнення та зручний інтерфейс.

Розроблюваний продукт має бути завершено згідно встановлених дедлайнів та доступних ресурсів. Програма має бути з адаптивним інтерфейсом, яка забезпечить автоматизацію основних робочих процесів.

2.1.2 Вимоги до персоналу

Від персоналу не має вимагатися особливих технічних навичок для підтримки й експлуатації десктоп-додатку, окрім загальних навичок роботи з персональним комп'ютером.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у десктоп-додатку повинна зберігатися у базі даних, реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Розроблюваний десктоп-додаток має бути доступним для співробітників організації. Кожен співробітник організації має свої дані для входу в систему (логін та пароль).

Права доступу до інформації розмежовані за групами користувачів: адміністратор, волонтер.

Адміністратор має найбільший за правами доступ серед всіх користувачів. Він має доступ до даних з правами перегляду, додавання, редагування та видалення інформації. Також адміністратор може додавати в систему нових працівників(волонтерів). Також є можливість ведення та аналізу звітності. Доступ до панелі адміністратора надається за спеціально зарезервованим логіном та паролем.

Волонтер має менший спектр доступу до інформації ніж адміністратор. До переліку його можливостей входять: перегляд, додавання, редагування та видалення інформації про замовників.

Доступ до програми здійснюється за унікальним логіном і паролем.

2.2 Структура декстоп-додатку

2.2.1 Загальна інформація про структуру десктоп-додатку

Структура десктоп-додатку являє собою сторінку, на якій доступні функції перегляду, додавання, редагування та видалення інформації. Пошук замовників за номером телефону, ПІБ. Створення, ведення та аналіз звітності.

Схематичний інтерфейс додатку представлена на рисунку А.1.

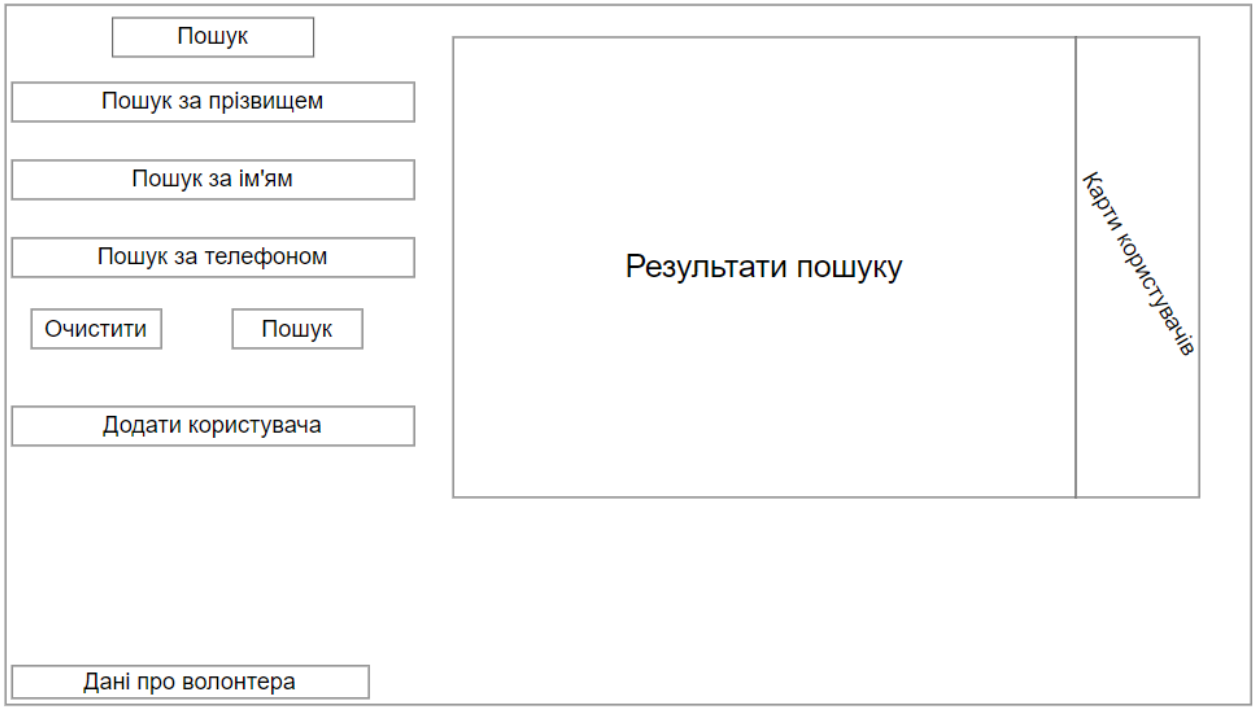


Рисунок А.1 – Схематичний інтерфейс додатку

2.2.2 Навігаційне меню

Для зручної навігації повинно бути створене меню, що забезпечить швидке переміщення волонтера по всім доступним функціям десктоп-додатку.

Меню має включати в собі функцію пошуку замовників за трьома ключовими параметрами, а саме прізвище, ім’я та/або номер телефону. Також повинна бути створена функція додавання нових замовників в систему,

відкриття існуючих карт замовників та можливість переглядати інформацію про волонтера.

2.2.3 Управління контентом

Заповнення та редагування контенту десктоп-додатку має бути зроблено через панель керування, використовуючи інформацію з бази даних.

Всю інформацію для наповнення десктоп-додатку має надавати адміністратор, включаючи всі анкетні дані та інформацію про волонтерську організацію.

2.2.4 Дизайн десктоп-додатку

Дизайн десктоп-додатку має бути виконаний у мінімалістичному та сучасному стилі. Корпоративними кольорами є білий та чорний колір. Тому під час розробки десктоп-додатку треба використовувати саме ці кольори.

Види і розміри шрифтів повинні бути комфортними для перегляду. Інформаційні блоки, графічні матеріали та інші елементи повинні мати зручне і логічне розташування.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Додавання інформації про замовників	Волонтер, Адміністратор
UN-02	Редагування інформації про замовників	Волонтер, Адміністратор
UN-03	Видалення інформації про замовників	Волонтер, Адміністратор
UN-04	Перегляд інформації про замовників	Волонтер, Адміністратор
UN-05	Додавання інформації про волонтерів	Адміністратор
UN-06	Редагування інформації про волонтерів	Адміністратор
UN-07	Видалення інформації про волонтерів	Адміністратор
UN-08	Створення, ведення та аналіз звітності	Адміністратор

2.3.2 Функціональні вимоги

На основі потреб волонтера були визначені такі функціональні вимоги:

- авторизація співробітників;
- реєстрація нових замовників;
- додавання/редагування/видалення інформації про замовників;

- перегляд інформації про замовників;
- пошук інформації про замовників;
- відображення результатів пошуку;
- додавання/редагування/видалення інформації про волонтерів;
- створення, ведення та аналіз звітності;

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація десктоп-додатку відбувається з використанням:

- C#
- MySQL 8.0

2.4.2 Вимоги до лінгвістичного забезпечення

Десктоп-додаток має бути виконаний українською та англійською мовами.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Операційна система: Windows 7 і вище.
- Центральний процесор: Intel Core i5 і вище.
- Оперативна пам’ять: 2 ГБ і вище.
- Відеокарта: Не є обов’язковою.
- Жорсткий диск: HDD/SSD 128 ГБ і вище.

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ ДЕСКТОП-ДОДАТКУ

Детальний опис етапів створення десктоп-додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення десктоп-додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Аналіз предметної області	2 дні
2	Складання технічного завдання	1 день
3	Підготовка прототипу	1 день
4	Створення макету дизайну десктоп-додатку	1 день
5	Розробка модулю реєстрації співробітників	3 дні
6	Розробка модулю реєстрації нових користувачів	5 днів
7	Розробка модулю перегляду/додавання/редагування/видалення інформації про замовників	5 днів
8	Розробка модулю перегляду/додавання/редагування/видалення інформації про волонтерів	3 дні
9	Розробка модулю перегляду та пошуку інформації про замовників	3 дні
10	Розробка модулю відображення результатів пошуку за параметрами	3 дні
11	Розробка модулю створення, ведення та аналізу звітності	4 дні
12	Beta-тестування	5 днів
13	Alpha-тестування	3 дні
14	Перевірка працездатності десктоп-додатку	1 день
15	Реліз програмного додатку	1 день
16	Створення програмної документації	5 днів
	Загальна тривалість робіт	46 днів

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВЕДЕННЯ ДЕСКТОП-ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Для того, щоб десктоп-додатком могли користуватися адміністратор та волонтери., необхідно створити установник програми. В подальшому співробітники встановлюють програму на свій ПК і починають користуватися нею через авторизацію в систему під своїм ім’ям.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Десктоп-додатки мають багато можливостей і становлять ефективне рішення для різних завдань.

Десктоп-додаток має бути розроблений у встановлений графіком час та з використанням наявних ресурсів [16]. Цей продукт буде актуальним тільки на території України, оскільки функціонал додатку є актуальним тільки для українців [17].

Результат деталізації даного проекту методом SMART розміщено у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific (конкретна)	Створення десктоп-додатку для автоматизації управління процесами у волонтерській організації
Measurable (вимірювана)	Оптимізація роботи волонтерської організації. Швидкий пошук та доступ до інформації.
Achievable (досяжна, узгоджена)	Мета досяжна, тому що є технічне завдання, воно затверджене, є попит на використання додатку та чіткий план виконання проекту.
Relevant (актуальна)	Актуальність пов’язана з потребою волонтерських організацій у недорогому програмному забезпеченні, яке спростить роботу для волонтерів, та створить зручність для ведення та аналізу звітності.
Time-framed (обмежена в часі)	Є конкретний термін – 10.06.2023

WBS – це графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту [18]. Верхній рівень WBS відображає продукт проекту, який повинен відповідати меті проекту. На другому рівні відображаються дії або основні етапи, необхідні для досягнення цілей проекту.

Розбивка цих етапів продовжується до елементарних робіт, які мають один чіткий результат – окреме виконання певної дії. На всі елементарні роботи призначений один відповідальний. WBS-структура представляє склад роботи та обчислює тривалість виконання кожної дії.

Кількість рівнів WBS може відрізнятися залежно від проекту. Для верхнього рівня зазвичай використовують іменники для позначення елементів, а на нижньому рівні - віддієслівні іменники. Елементарні роботи можуть бути ще більш деталізовані, але це вимагає спеціалізованих знань.

Важливо розуміти, що елементарні роботи нижнього рівня WBS не є деталізованими з погляду управління проектами, але насправді вони можуть бути ще більш дрібними і можуть бути розділені на окремі операції. Проте для цього потрібні спеціалізовані технологічні знання, які можуть бути отримані на місці виконання роботи. Діаграму WBS представлено на рисунку Б.1

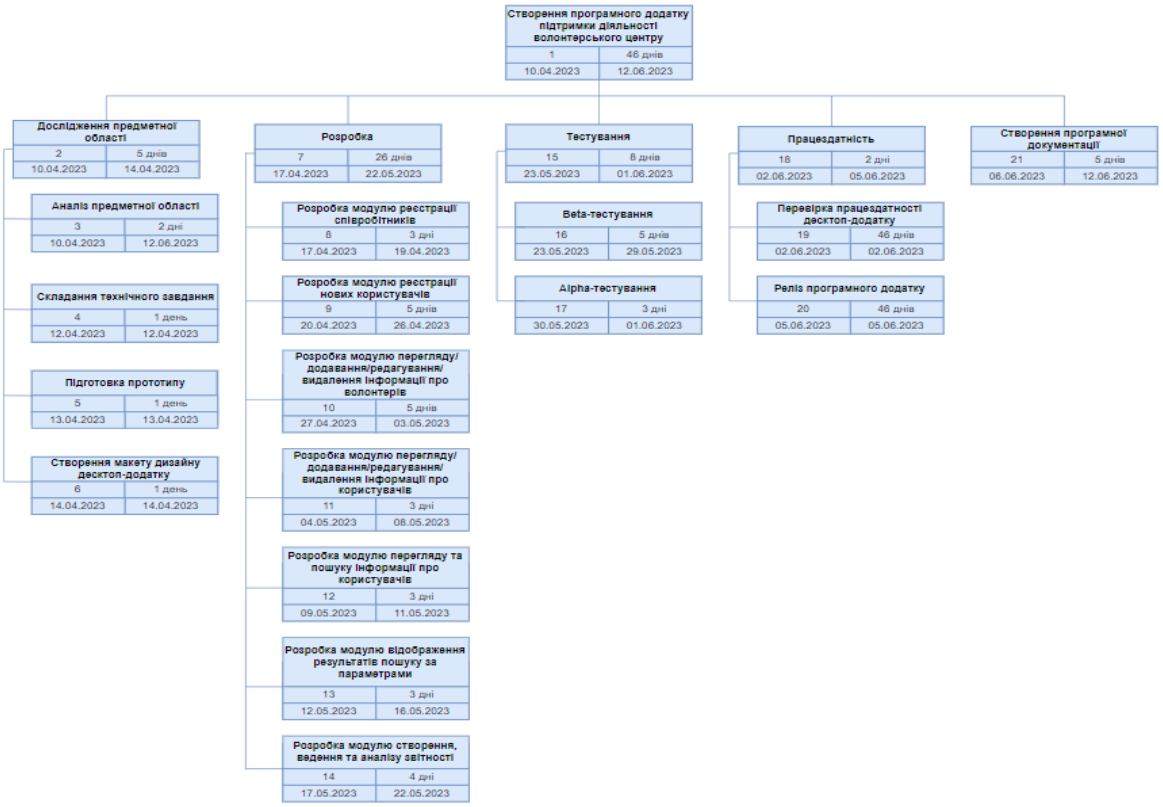


Рисунок Б.1 – Планування змісту структури роботи за допомогою діаграми WBS

Після розбиття процесів на складові етапи, наступним кроком є розробка графічної структури, що відображає учасників або відповідальних осіб, які беруть участь у проекті, так звану організаційну структуру виконавців (OBS). Відповідальні особи в цій структурі - співробітники, які забезпечують організацію та виконання елементарних робіт, які вказані у WBS. Кожна з цих елементарних робіт може бути розглянута як окремий проект. На рисунку Б.2 показана організаційна структура для планування проекту.

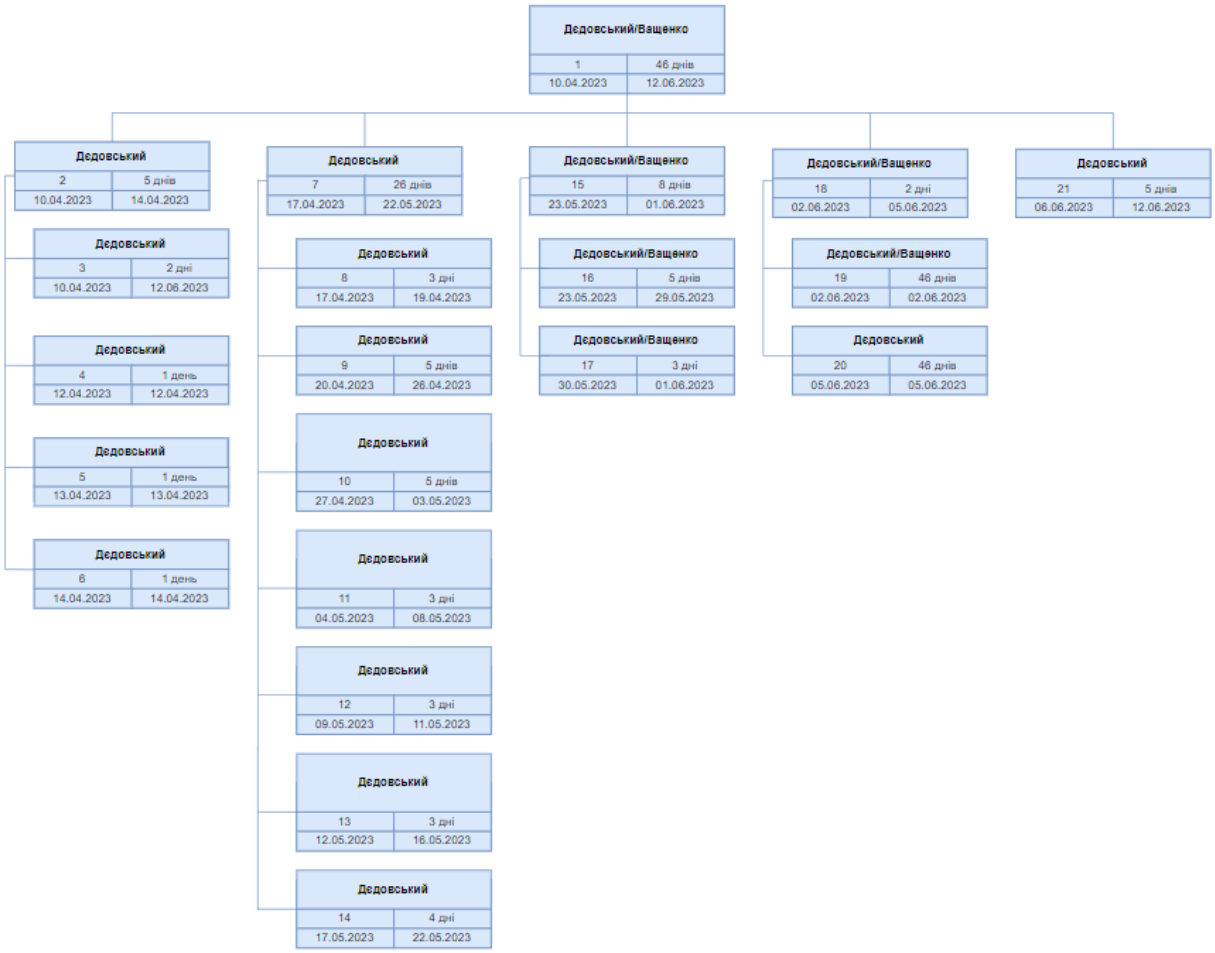
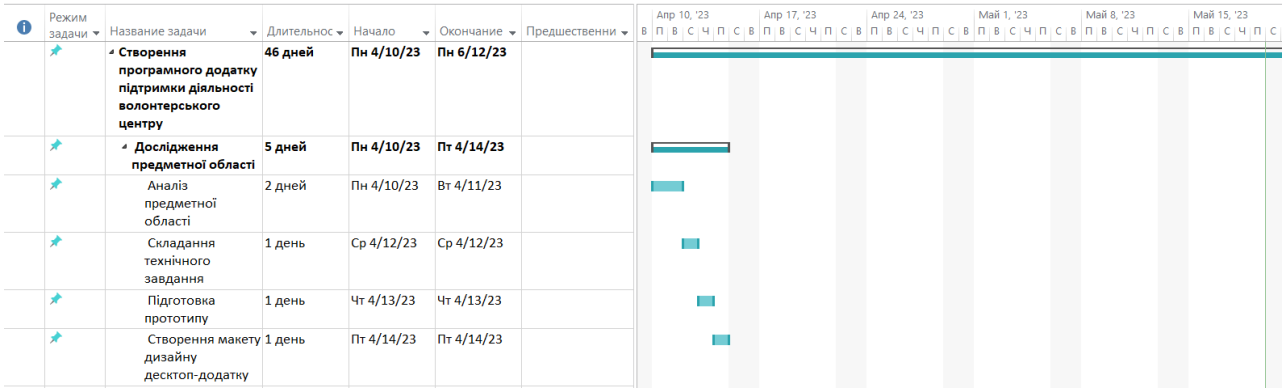


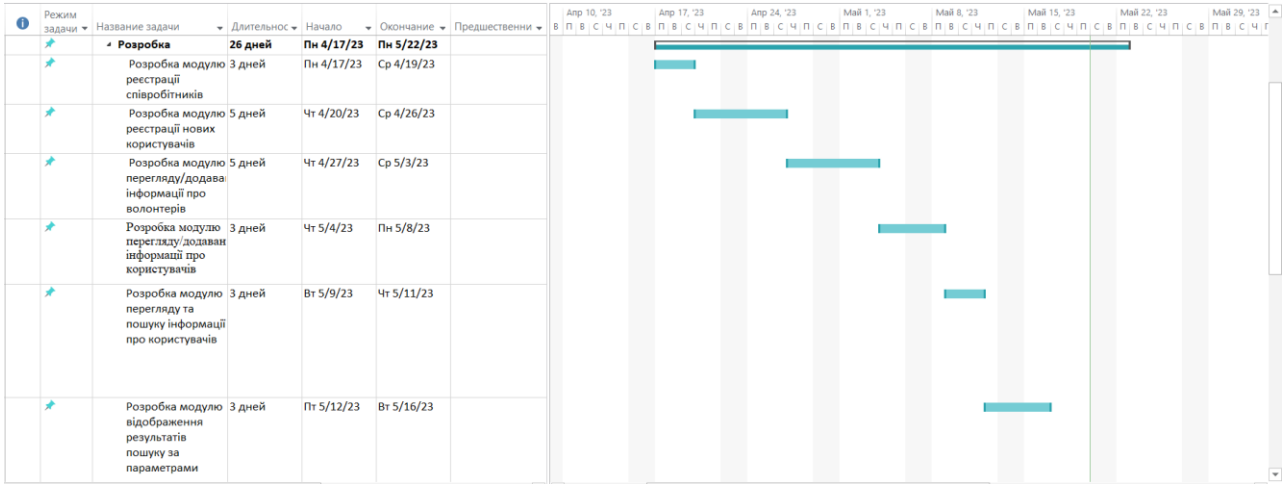
Рисунок Б.2 – OBS-структура проекту

Один з ключових етапів планування проекту - побудова календарного графіку (діаграми Ганта), який показує розклад виконання робіт з урахуванням реальних дат [19]. Це дозволяє отримати точне уявлення про терміни виконання процесів з врахуванням обмежень в ресурсах, вихідних днів та свят. Календарний графік проекту можна побачити на рисунках Б.3-Б.5



Режим задачи	Название задачи	Длительнос	Начало	Окончание	Предшественни
	Створення програмного додатку підтримки діяльності волонтерського центру	**46 дней**	**Пн 4/10/23**	**Пн 6/12/23**	
	Дослідження предметної області	**5 дней**	**Пн 4/10/23**	**Пт 4/14/23**	
	Аналіз предметної області	2 дней	Пн 4/10/23	Вт 4/11/23	
	Складання технічного завдання	1 день	Ср 4/12/23	Ср 4/12/23	
	Підготовка прототипу	1 день	Чт 4/13/23	Чт 4/13/23	
	Створення макету дизайну десктоп-додатку	1 день	Пт 4/14/23	Пт 4/14/23	

Рисунок Б.3 – Загальний вигляд діаграми всього проекту, частина 1



Режим задачи	Название задачи	Длительнос	Начало	Окончание	Предшественни
	Розробка	**26 дней**	**Пн 4/17/23**	**Пн 5/22/23**	
	Розробка модулю реєстрації співробітників	3 дней	Пн 4/17/23	Ср 4/19/23	
	Розробка модулю реєстрації нових користувачів	5 дней	Чт 4/20/23	Ср 4/26/23	
	Розробка модулю перегляду/додавання інформації про волонтерів	5 дней	Чт 4/27/23	Ср 5/3/23	
	Розробка модулю перегляду/додавання інформації про користувачів	3 дней	Чт 5/4/23	Пн 5/8/23	
	Розробка модулю перегляду та пошуку інформації про користувачів	3 дней	Вт 5/9/23	Чт 5/11/23	
	Розробка модулю відображення результатів пошуку за параметрами	3 дней	Пт 5/12/23	Вт 5/16/23	

Рисунок Б.4 – Загальний вигляд діаграми всього проекту, частина

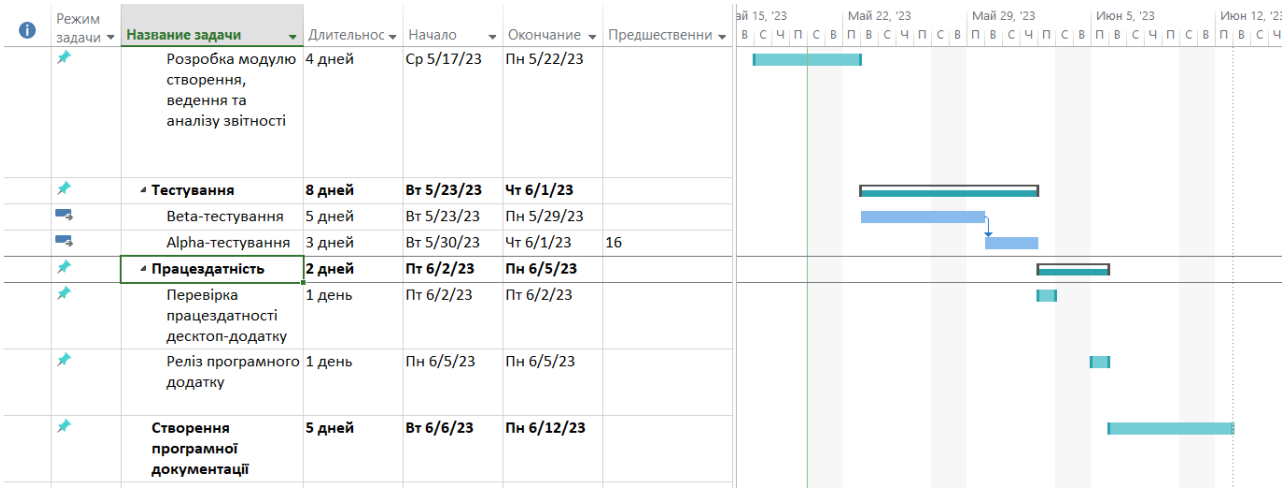


Рисунок Б.5 – Загальний вигляд діаграми всього проекту, частина 3

Під час проведення якісної оцінки ризиків необхідно виділити ті, які потребують негайної уваги. Реакція на кожний ризик буде залежати від його значущості. Після цього, наступним кроком є кількісна оцінка ризиків [20]. Якісна та кількісна оцінки можуть виконуватися окремо або одночасно, залежно від рівня забезпечення проекту. Шкала класифікації ризиків за їх впливом на проект та ймовірністю виникнення подана в таблиці Б.3.

Таблиця Б.3 – Шкала класифікації ризиків за ймовірністю виникнення та величиною впливу на проект

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Щоб зменшити негативний вплив ризиків на проект, необхідно здійснити планування заходів по їхньому усуненню. Це включає визначення ефективності заходів та оцінку наслідків, які можуть впливати на проект. Оцінка проводиться згідно з показниками, які наведені в таблиці Б.3. У результаті планування була створена матриця ймовірності виникнення ризиків та їхнього впливу, яку можна побачити в таблицях Б.4.-Б.5. Зеленим кольором

позначають прийнятні ризики, жовтим - виправдані, а червоним – неприпустимі.

Таблиця Б.4 – Матриця ймовірності та впливу

Ймовірність ризику(Й)	Вплив загрози(ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0,05	0,1	0,2	0,4	0,8
0,9			R1(0,18)		
0,7			R3(0,14)		
0,5			R6(0,1)		R7(0,4)
0,3				R2(0,12)	
0,1				R4(0,04), R5(0,04)	R8(0,08)

Таблиця Б.4 містить класифікацію ризиків за рівнем, відповідно до отриманого значення індексу. У таблицях Б.5.-Б.6. надано опис ризиків та відповідні стратегії реагування на кожен з них.

Таблиця Б.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$0,005 \leq R \leq 0,05$	4,5
2	Виправдані	$0,05 < R \leq 0,14$	2,3,6,8
3	Недопустимі	$0,14 < R \leq 0,72$	1,7

Таблиця Б.6 – Стратегії реагування на ризики

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив	Ранг ризику	План А	Тип стратегії	План Б
RS_1	Відкритий	Частина роботи яка викликає складність виконання	Малий	Великий	0,18	Отримати знання по ходу виконання	Ухилення	Негайна консультація з викладачем
RS_2	Відкритий	Технічні збої(відсутність електроенергії, доступу до Інтернету)	Малий	Середній	0,12	Мати резервне джерело живлення	Ухилення	Використати техніку в іншому місці
RS_3	Відкритий	Зміна ТЗ	Середній	Середній	0,14	Домовитись про зміни	Зменшення	Внести зміни в ТЗ
RS_4	Відкритий	Погане самопочуття виконавця	Малий	Середній	0,04	Негайно відвідати лікаря	Попередження	Виконувати регулярно невеликий обсяг роботи
RS_5	Відкритий	Знаходження помилок під час тестування	Середній	Середній	0,04	Зробити відповідні зміни, внести їх в ТЗ	Ухилення	Обговорити з керівником

Таблиця Б.7 – Продовження таблиці Б.6

RS_6	Відкритий	Затримка розробки по календарному графіку	Середній	Великий	0,1	Пришвидшити роботу за рахунок зменшення розробки додаткових функцій	Попередження	Обговорити з керівником шляхи вирішення проблеми
RS_7	Відкритий	Помилки при написанні документації	Малий	Середній	0,4	Зробити відповідні зміни в ТЗ	Зменшення	
RS_8	Відкритий	Втрата файлів	Малий	Великий	0,08	Робити резервні копії, використати саме їх	Попередження	Віднови ти втрачені файли за допомогою спеціальних програм

ДОДАТОК В

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ ДЕСКТОП ДОДАТКУ

AddCustomerForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
   public partial class AddCustomerForm : Form
   {
      public AddCustomerForm()
      {
         InitializeComponent();
      }

      private void textBox2_TextChanged(object sender, EventArgs e)
      {

      }

      private void AddCustomerForm_Load(object sender, EventArgs e)
      {
```

```
}

private void textBoxPhone_Enter(object sender, EventArgs e)
{
   if(textBoxPhone.Text == "0957312972")
   {
      textBoxPhone.Text = "";

      textBoxPhone.ForeColor = Color.Black;
   }
}

private void textBoxPhone_Leave(object sender, EventArgs e)
{
   if (textBoxPhone.Text == "")
   {
      textBoxPhone.Text = "0957312972";

      textBoxPhone.ForeColor = Color.Silver;
   }
}

private void textBoxBirthDate_Enter(object sender, EventArgs e)
{
   if (textBoxBirthDate.Text == "2002-12-31")
   {
      textBoxBirthDate.Text = "";

      textBoxPhone.ForeColor = Color.Black;
   }
}

private void textBoxBirthDate_Leave(object sender, EventArgs e)
{
   if (textBoxBirthDate.Text == "")
   {
      textBoxBirthDate.Text = "2002-12-31";
```

```
        textBoxPhone.ForeColor = Color.Silver;
    }
}

private void buttonAdd_Click(object sender, EventArgs e)
{
    if (!IsDigitsOnly(textBoxPhone.Text))
    {
        MessageBox.Show("Перевірте номер телефону");
        return;
    }
    if (textBoxGender.Text != "Ч" && textBoxGender.Text != "Ж")
    {
        MessageBox.Show("Перевірте поле стать. Правильно (Ч) або (Ж)!!!");
        return;
    }
    if (IsDigitsOnly(textBoxSurname.Text))
    {
        MessageBox.Show("Перевірте прізвище");
        return;
    }
    if (IsDigitsOnly(textBoxName.Text))
    {
        MessageBox.Show("Перевірте ім'я");
        return;
    }
    MySQL mysql = new MySQL();
    try
    {
        mysql.OpenConnection();
    }
    catch
    {
        MessageBox.Show("Проблеми з доступом до бази даних!!");
        return;
    }
```

```
            string s = textBoxSurname.Text;
            string n = textBoxName.Text;
            string p = textBoxPhone.Text;
            string bc = textBoxBirthCity.Text;
            string bs = textBoxBirthStreet.Text;
            string bh = textBoxBirthHouse.Text;
            string fc = textBoxFactCity.Text;
            string g = textBoxGender.Text;
            DateTime d = Convert.ToDateTime(textBoxBirthDate.Text);
            string c = textBoxComment.Text;

            MySqlCommand command = new MySqlCommand("INSERT INTO customertable (id, surname, name, phone, birthcity, birthstreet, birthhouse, factcity, gender, birthdate, comment) " +
                "VALUES (NULL, @s, @n, @p, @bc, @bs, @bh, @fc, @g, @bd, @c);", mysql.GetConnection());
            command.Parameters.Add("@s", MySqlDbType.VarChar).Value = s;
            command.Parameters.Add("@n", MySqlDbType.VarChar).Value = n;
            command.Parameters.Add("@p", MySqlDbType.VarChar).Value = p;
            command.Parameters.Add("@bc", MySqlDbType.VarChar).Value = bc;
            command.Parameters.Add("@bs", MySqlDbType.VarChar).Value = bs;
            command.Parameters.Add("@bh", MySqlDbType.VarChar).Value = bh;
            command.Parameters.Add("@fc", MySqlDbType.VarChar).Value = fc;
            command.Parameters.Add("@g", MySqlDbType.VarChar).Value = g;
            command.Parameters.Add("@bd", MySqlDbType.Date).Value = d.ToString("yyyy-MM-dd");
            command.Parameters.Add("@c", MySqlDbType.VarChar).Value = c;
            command.ExecuteNonQuery();

            mysql.CloseConnection();

            MessageBox.Show("Додано користувача");
            this.Close();
        }

        private void buttonClear_Click(object sender, EventArgs e)
        {
            textBoxSurname.Text = "";
            textBoxName.Text = "";
            textBoxBirthCity.Text = "";
```

```
            textBoxBirthStreet.Text = "";
            textBoxBirthHouse.Text = "";
            textBoxFactCity.Text = "";
            textBoxGender.Text = "";
            textBoxComment.Text = "";
        }
        public bool IsDigitsOnly(string str)
        {
            foreach (char c in str)
            {
                if (c < '0' || c > '9')
                    return false;
            }
            return true;
        }
    }
}
```

AddDeliveryForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class AddDeliveryForm : Form
    {
        public string Id_C { get; set; }
        public string Id_V { get; set; }
        public string Surname_C { get; set; }
```

```
public string Name_C { get; set; }
public string Surname_V { get; set; }
public string Name_V { get; set; }
public string pack { get; set; }
public int amount_pack { get; set; }
public AddDeliveryForm()
{
   InitializeComponent();
}
public AddDeliveryForm(string id_v, string id_c, string sur_v, string nam_v, string sur_c, string nam_c)
{
   Id_C = id_c;
   Id_V = id_v;
   Surname_C = sur_c;
   Surname_V = sur_v;
   Name_C = nam_c;
   Name_V = nam_v;
   InitializeComponent();
}
private void AddDeliveryForm_Load(object sender, EventArgs e)
{
   labelCustomerShow.Text = Surname_C + " " + Name_C;
   labelVolunteerShow.Text = Surname_V + " " + Name_V;
   labelDateShow.Text = DateTime.Now.ToString("dd/MM/yyyy");
   ComboBoxShow();
}
// Відображення вмісту доступних наборів в комбобоксі
public void ComboBoxShow()
{
   MySQL mysql = new MySQL();
   try
   {
      mysql.OpenConnection();
   }
   catch
   {
      MessageBox.Show("Проблеми з доступом до бази даних!!");
      this.Close();
```

```
            return;
        }

        MySqlCommand command = new MySqlCommand("SELECT id, name FROM packagetable",
mysql.GetConnection());
        MySqlDataAdapter da = new MySqlDataAdapter();
        da.SelectCommand = command;
        DataTable table1 = new DataTable();
        da.Fill(table1);

        DataRow itemrow = table1.NewRow();
        itemrow[1] = "Оберіть видачу";
        table1.Rows.InsertAt(itemrow, 0);

        comboBoxPackageShow.DataSource = table1;
        comboBoxPackageShow.DisplayMember = "name";
        comboBoxPackageShow.ValueMember = "id";

        mysql.CloseConnection();
    }
    private void comboBoxPackageShow_SelectedValueChanged(object sender, EventArgs e)
    {
        pack = comboBoxPackageShow.SelectedIndex.ToString();
    }
    private void buttonSavePackage_Click(object sender, EventArgs e)
    {
        if (comboBoxPackageShow.SelectedIndex.ToString() == "0")
        {
            MessageBox.Show("Ви не обрали видачу");
            return;
        }
        MySQL mysql = new MySQL();
        try
        {
            mysql.OpenConnection();
        }
        catch
        {
```

```
            MessageBox.Show("Проблеми з доступом до бази даних!!");
            return;
        }

        string EnterCm = richTextBoxCommentShow.Text;
        int EnterC = Convert.ToInt32(Id_C);
        int EnterV = Convert.ToInt32(Id_V);
        int EnterP = Convert.ToInt32(pack);

        MySqlCommand command = new MySqlCommand("SELECT amount FROM packagetable WHERE id = @id", mysql.GetConnection());
        command.Parameters.Add("@id", MySqlDbType.Int32).Value = pack;
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                amount_pack = Convert.ToInt32(reader["amount"]);
            }
        }

        if (amount_pack == 0)
        {
            MessageBox.Show("Немає в наявності даного пакунку, оберіть інший");
        }
        else
        {
            amount_pack--;
            MessageBox.Show("кількість наборів = " + amount_pack);

            command = new MySqlCommand("INSERT INTO deliverytable (id, date, customerid, volunteerid, packageid, comment) VALUES (NULL, @uD, @uC, @uV, @uP, @uCm);", mysql.GetConnection());
            command.Parameters.Add("@uD", MySqlDbType.Date).Value = DateTime.Now.ToString("yyyy/MM/dd");
            command.Parameters.Add("@uC", MySqlDbType.Int32).Value = EnterC;
            command.Parameters.Add("@uV", MySqlDbType.Int32).Value = EnterV;
            command.Parameters.Add("@uP", MySqlDbType.Int32).Value = EnterP;
            command.Parameters.Add("@uCm", MySqlDbType.VarChar).Value = EnterCm;
            command.ExecuteNonQuery();
```

```
                command = new MySqlCommand("UPDATE packagetable SET amount = @amo WHERE id =
@id", mysql.GetConnection());
                command.Parameters.Add("@amo", MySqlDbType.Int32).Value = amount_pack;
                command.Parameters.Add("@id", MySqlDbType.Int32).Value = pack;
                command.ExecuteNonQuery();
                mysql.CloseConnection();
            }
        }
        // Повернутися
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
            ShowCustomCardForm NewForm = new ShowCustomCardForm(Id_V, Id_C, Surname_V, Name_V,
Surname_C, Name_C);
            NewForm.Show();
        }
    }
}
```

AddPackageForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class AddPackageForm : Form
    {
```

```
public AddPackageForm()
{
   InitializeComponent();
}

private void AddPackageForm_Load(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
   if (!IsDigitsOnly(textBoxAmountPack.Text))
   {
      MessageBox.Show("Перевірте кількість пакунків");
      return;
   }
   MySQL mysql = new MySQL();
   try
   {
      mysql.OpenConnection();
   }
   catch
   {
      MessageBox.Show("Проблеми з доступом до бази даних!!");
      return;
   }

   string n = textBoxNamePack.Text;
   string a = textBoxAmountPack.Text;
   string c = richTextBoxComment.Text;


   MySqlCommand command = new MySqlCommand("INSERT INTO packagetable (id, name, amount, comment) " +
      "VALUES (NULL, @n, @a, @c);", mysql.GetConnection());
   command.Parameters.Add("@n", MySqlDbType.VarChar).Value = n;
   command.Parameters.Add("@a", MySqlDbType.VarChar).Value = a;
```

```
            command.Parameters.Add("@c", MySqlDbType.VarChar).Value = c;

            command.ExecuteNonQuery();

            mysql.CloseConnection();

            MessageBox.Show("Додано пакунок");
            this.Close();
        }
        public bool IsDigitsOnly(string str)
        {
            foreach (char c in str)
            {
                if (c < '0' || c > '9')
                    return false;
            }
            return true;
        }
    }
}
```

AddVolunteerForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class AddVolunteerForm : Form
```

```
{
    public AddVolunteerForm()
    {
        InitializeComponent();
    }
    private void textBoxBirthDate_Enter(object sender, EventArgs e)
    {
        if (textBoxBirthDate.Text == "1999-12-25")
        {
            textBoxBirthDate.Text = "";

            textBoxBirthDate.ForeColor = Color.Black;
        }
    }

    private void textBoxBirthDate_Leave(object sender, EventArgs e)
    {
        if (textBoxBirthDate.Text == "")
        {
            textBoxBirthDate.Text = "1999-12-25";

            textBoxBirthDate.ForeColor = Color.Silver;
        }
    }

    private void textBoxPhone_Enter(object sender, EventArgs e)
    {
        if (textBoxPhone.Text == "0957312972")
        {
            textBoxPhone.Text = "";

            textBoxPhone.ForeColor = Color.Black;
        }
    }

    private void textBoxPhone_Leave(object sender, EventArgs e)
    {
        if (textBoxPhone.Text == "")
```

```
    {
        textBoxPhone.Text = "0957312972";

        textBoxPhone.ForeColor = Color.Silver;
    }
}

private void buttonAdd_Click(object sender, EventArgs e)
{
    if (IsDigitsOnly(textBoxSurname.Text))
    {
        MessageBox.Show("Перевірте прізвище");
        return;
    }
    if (IsDigitsOnly(textBoxName.Text))
    {
        MessageBox.Show("Перевірте ім'я");
        return;
    }
    if (!IsDigitsOnly(textBoxPhone.Text))
    {
        MessageBox.Show("Перевірте номер телефону");
        return;
    }
    if (IsDigitsOnly(textBoxCity.Text))
    {
        MessageBox.Show("Перевірте місто");
        return;
    }

    if (textBoxPassword.Text != textBoxPasswordAgain.Text)
    {
        MessageBox.Show("Паролі не збігаються");
        return;
    }

    MySQL mysql = new MySQL();
    try
```

```
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                return;
            }

            string s = textBoxSurname.Text;
            string n = textBoxName.Text;
            DateTime bd = Convert.ToDateTime(textBoxBirthDate.Text);
            string p = textBoxPhone.Text;
            string c = textBoxCity.Text;
            string l = textBoxLogin.Text;
            string pw = textBoxPassword.Text;
            string pwa = textBoxPasswordAgain.Text;



            MySqlCommand command = new MySqlCommand("INSERT INTO volunteertable (id, surname, name, birthdate, phone, factcity, login, password) " +
                "VALUES (NULL, @s, @n, @bd, @p, @c, @l, @pw);", mysql.GetConnection());
            command.Parameters.Add("@s", MySqlDbType.VarChar).Value = s;
            command.Parameters.Add("@n", MySqlDbType.VarChar).Value = n;
            command.Parameters.Add("@bd", MySqlDbType.Date).Value = bd.ToString("yyyy-MM-dd");
            command.Parameters.Add("@p", MySqlDbType.VarChar).Value = p;
            command.Parameters.Add("@c", MySqlDbType.VarChar).Value = c;
            command.Parameters.Add("@l", MySqlDbType.VarChar).Value = l;
            command.Parameters.Add("@pw", MySqlDbType.VarChar).Value = pw;
            command.ExecuteNonQuery();

            mysql.CloseConnection();

            MessageBox.Show("Додано користувача");
           //  this.Close();
        }
        public bool IsDigitsOnly(string str)
```

```
        {
            foreach (char c in str)
            {
                if (c < '0' || c > '9')
                    return false;
            }
            return true;
        }

        private void buttonReturn_Click(object sender, EventArgs e)
        {
            VolunteerListForm NewForm = new VolunteerListForm();
            NewForm.Show();
            this.Close();
        }
    }
}
```

AdminPanel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class AdminPanel : Form
    {
        public AdminPanel()
        {
            InitializeComponent();
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            VolunteerListForm NewForm = new VolunteerListForm();
            NewForm.Show();
        }

        private void PackageListbutton_Click(object sender, EventArgs e)
        {
            DeliveryForm NewForm = new DeliveryForm();
            NewForm.Show();
        }

        private void Statbutton_Click(object sender, EventArgs e)
        {
            DeliveryStatForm NewForm = new DeliveryStatForm();
            NewForm.Show();
        }
    }
}
```

Customer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraduateWork
{
    public class Customer
    {
        public int Id { get; set; }
        public string Surname { get; set; }
        public string Name { get; set; }
```

```
        public int Phone { get; set; }
        public string BirthCity { get; set; }
        public string BirthStreet { get; set; }
        public string BirthHouse { get; set; }
        public string FactCity { get; set; }
        public string Gender { get; set; }
        public DateTime BirthDate { get; set; }
        public string Comment { get; set; }

        public Customer(string surname, string name, int phone, string birthcity, string birthstreet, string birthhouse, string factcity, string gender, DateTime birthdate, string comment)
        {
            Surname = surname;
            Name = name;
            Phone = phone;
            BirthCity = birthcity;
            BirthStreet = birthstreet;
            BirthHouse = birthhouse;
            FactCity = factcity;
            Gender = gender;
            BirthDate = birthdate;
            Comment = comment;
        }
        public Customer(int id, string surname, string name, int phone, string birthcity, string factcity)
        {
            Id = id;
            Surname = surname;
            Name = name;
            Phone = phone;
            BirthCity = birthcity;
            FactCity = factcity;
        }


    }
}
```

Delivery.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraduateWork
{
   public class Delivery
   {
      public int Id { get; set; }
      public DateTime Date { get; set; }
      public int Id_Customer { get; set; }
      public int Id_Volunteer { get; set; }
      public int Id_Package { get; set; }
      public string Name_Package { get; set; }
      public string FullName_Volunteer { get; set; }
      public string FullName_Customer { get; set; }
      public string CustomerPhone { get; set; }
      public string CustomerCity { get; set; }
      public string Comment { get; set; }

      public Delivery(int id, DateTime date, int id_c, int id_v, int id_p, string comment)
      {
         Id = id;
         Date = date;
         Id_Customer = id_c;
         Id_Volunteer = id_v;
         Id_Package = id_p;
         Comment = comment;
      }
      public Delivery(int id, DateTime date, int id_c, int id_v, string name_p, string fullname_v)
      {
         Id = id;
         Date = date;
         Id_Customer = id_c;
```

```
            Id_Volunteer = id_v;
            Name_Package = name_p;
            FullName_Volunteer = fullname_v;
        }
        public Delivery(int id, DateTime date, string name_p, string fullname_v)
        {
            Id = id;
            Date = date;
            Name_Package = name_p;
            FullName_Volunteer = fullname_v;
        }
        public Delivery(DateTime date, string name_p, string fullname_c, string phone_c, string city_c, string
fullname_v)
        {
            Date = date;
            Name_Package = name_p;
            FullName_Customer = fullname_c;
            CustomerPhone = phone_c;
            CustomerCity = city_c;
            FullName_Volunteer = fullname_v;
        }
    }
}
```

DeliveryForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Xceed.Document.NET;
```

```
using Xceed.Words.NET;

namespace GraduateWork
{
   public partial class DeliveryForm : Form
   {
      public DeliveryForm()
      {
         InitializeComponent();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         string WordDocumentPath = "СписокПакунків.docx";
         DocX document = DocX.Create(WordDocumentPath);

         Paragraph paragraph1 = document.InsertParagraph();
         paragraph1.AppendLine("Список Пакунків: ")
            .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
            .FontSize(16);
         Table t = document.AddTable(dataGridView1.Rows.Count + 1, dataGridView1.Columns.Count);
         t.Alignment = Alignment.left;
         t.Design = TableDesign.ColorfulListAccent5;

         for (int i = 0; i < 3; i++)
         {
            t.Rows[0].Cells[i].Paragraphs[0].Append(dataGridView1.Columns[i].HeaderText)
               .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
               .FontSize(14)
               .Bold();
         }
         for (int i = 1; i < t.Rows.Count; i++)
         {
            for (int j = 0; j < 4; j++)
            {
               object SomeVal = dataGridView1[j, i - 1].Value;
               if (SomeVal != null)
               {
```

```
                t.Rows[i].Cells[j].Paragraphs[0].Append(SomeVal.ToString())
                    .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                    .FontSize(10);
            }
        }
    }
    document.InsertTable(t);

    Paragraph paragraph2 = document.InsertParagraph();
    document.Save();
    MessageBox.Show("Звіт створений. Назва файлу — СписокПакунків.docx");
}

private void button1_Click(object sender, EventArgs e)
{
    AddPackageForm NewForm = new AddPackageForm();
    NewForm.Show();
    this.Close();
}

private void DeliveryForm_Load(object sender, EventArgs e)
{
    List<Package> _pack = GetPack();
    if (_pack.Count == 0)
    {
        MessageBox.Show("Клієнта не знайдено");
        //  ClearTextBox();
    }
    else
    {
        foreach (Package p in _pack)
        {
            dataGridView1.Rows.Add(p.Id, p.Name, p.Amount, p.Comment);
        }
        foreach (DataGridViewRow r in dataGridView1.Rows)
        {
            DataGridViewButtonCell ButtonCell = (DataGridViewButtonCell)r.Cells[4];
            ButtonCell.Value = "Редагувати";
```

```
        }
        dataGridView1.ReadOnly = true;
    }
}

public List<Package> GetPack()
{
    MySQL mysql = new MySQL();
    try
    {
        mysql.OpenConnection();
    }
    catch
    {
        MessageBox.Show("Проблеми з доступом до бази даних!!");
        this.Close();
        return null;
    }
    String request = "SELECT * FROM packagetable";
    MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
    command.ExecuteNonQuery();
    List<Package> p = new List<Package>();
    using (MySqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            p.Add(new Package(Convert.ToString(reader["id"]),
                Convert.ToString(reader["name"]),
                Convert.ToInt32(reader["amount"]),
                Convert.ToString(reader["comment"])));
        }
    }
    mysql.CloseConnection();
    return p;
}

private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
```

```
        {
            if (e.ColumnIndex == 4)
            {
                int u = e.RowIndex;
                string id = dataGridView1.Rows[u].Cells[0].Value.ToString();
                MovePackageForm NewForm = new MovePackageForm(id);
                NewForm.Show();
            }
        }
    }
}
```

DeliveryStatForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Xceed.Document.NET;
using Xceed.Words.NET;

namespace GraduateWork
{
    public partial class DeliveryStatForm : Form
    {
        public DeliveryStatForm()
        {
            InitializeComponent();
        }

        private void DeliveryStatForm_Load(object sender, EventArgs e)
```

```
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return;
            }

            MySqlCommand command = new MySqlCommand("SELECT id, name FROM packagetable",
mysql.GetConnection());
            MySqlDataAdapter da = new MySqlDataAdapter();
            da.SelectCommand = command;
            DataTable table1 = new DataTable();
            da.Fill(table1);

            DataRow itemrow = table1.NewRow();
            itemrow[1] = "Всі пакунки";
            table1.Rows.InsertAt(itemrow, 0);

            comboBoxPackageShow.DataSource = table1;
            comboBoxPackageShow.DisplayMember = "name";
            comboBoxPackageShow.ValueMember = "id";

            mysql.CloseConnection();
        }

        public List<Delivery> GetDeliver()
        {
            string bdate = dateTimePickerBegin.Value.ToString("yyyy-MM-dd");
            string edate = dateTimePickerEnd.Value.ToString("yyyy-MM-dd");
            MySQL mysql = new MySQL();
            try
            {
```

```
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return null;
            }
            string pack = comboBoxPackageShow.SelectedIndex.ToString();
            String request = "SELECT dt.date, pt.name, CONCAT_WS(' ', ct.surname, ct.name) AS cname,
ct.phone, " +
                "ct.birthcity, CONCAT_WS(' ', vt.surname, vt.name) AS vname FROM deliverytable dt " +
                "JOIN packagetable pt ON dt.packageid = pt.id JOIN customertable ct ON dt.customerid = ct.id "
+
                "JOIN volunteertable vt ON dt.volunteerid = vt.id WHERE dt.date >= @bdate AND dt.date <=
@edate"; ;
            if (pack != "0")
            {
                request = "SELECT dt.date, pt.name, CONCAT_WS(' ', ct.surname, ct.name) AS cname, ct.phone,
" +
                "ct.birthcity, CONCAT_WS(' ', vt.surname, vt.name) AS vname FROM deliverytable dt " +
                "JOIN packagetable pt ON dt.packageid = pt.id JOIN customertable ct ON dt.customerid = ct.id "
+
                "JOIN volunteertable vt ON dt.volunteerid = vt.id WHERE dt.date >= @bdate AND dt.date <=
@edate AND pt.id = @packname";
            }
            MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
            command.Parameters.AddWithValue("@bdate", bdate);
            command.Parameters.AddWithValue("@edate", edate);
            command.Parameters.AddWithValue("@packname", pack);
            command.ExecuteNonQuery();
            List<Delivery> d = new List<Delivery>();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    d.Add(new Delivery(Convert.ToDateTime(reader["date"]),
                        Convert.ToString(reader["name"]),
```

```
                    Convert.ToString(reader["cname"]),
                    Convert.ToString(reader["phone"]),
                    Convert.ToString(reader["birthcity"]),
                    Convert.ToString(reader["vname"])));
                }
            }
            mysql.CloseConnection();
            return d;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView.Rows.Clear();
            List<Delivery> _del = GetDeliver();
            if (_del.Count == 0)
            {
                MessageBox.Show("Видач за даний період не знайдено");
                //  ClearTextBox();
            }
            else
            {
                foreach (Delivery d in _del)
                {
                    dataGridView.Rows.Add(d.Date.ToString("dd/MM/yyyy"),                    d.Name_Package,
d.FullName_Customer, d.CustomerPhone, d.CustomerCity, d.FullName_Volunteer);
                }
                dataGridView.ReadOnly = true;
            }

            labelAmount.Text = "Кількість записів за цей період: " + dataGridView.Rows.Count;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string WordDocumentPath = "СписокВидач.docx";
            DocX document = DocX.Create(WordDocumentPath);

            Paragraph paragraph1 = document.InsertParagraph();
```

```
                paragraph1.AppendLine("Кількість Видач за період з " +
dateTimePickerBegin.Value.ToString("dd/MM/yyyy")
                    + " по " + dateTimePickerEnd.Value.ToString("dd/MM/yyyy") + ": " + dataGridView.Rows.Count)
                    .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                    .FontSize(16);
                paragraph1.AppendLine("Список Видач за період з " +
dateTimePickerBegin.Value.ToString("dd/MM/yyyy")
                    + " по " + dateTimePickerEnd.Value.ToString("dd/MM/yyyy")  + ": ")
                    .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                    .FontSize(16);
                Table t = document.AddTable(dataGridView.Rows.Count + 1, dataGridView.Columns.Count);
                t.Alignment = Alignment.left;
                t.Design = TableDesign.ColorfulListAccent5;

                for (int i = 0; i < 6; i++)
                {
                    t.Rows[0].Cells[i].Paragraphs[0].Append(dataGridView.Columns[i].HeaderText)
                        .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                        .FontSize(14)
                        .Bold();
                }
                for (int i = 1; i < t.Rows.Count; i++)
                {
                    for (int j = 0; j < 6; j++)
                    {
                        object SomeVal = dataGridView[j, i - 1].Value;
                        if (SomeVal != null)
                        {
                            t.Rows[i].Cells[j].Paragraphs[0].Append(SomeVal.ToString())
                                .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                                .FontSize(10);
                        }
                    }
                }
                document.InsertTable(t);

                Paragraph paragraph2 = document.InsertParagraph();
                document.Save();
```

```
            MessageBox.Show("Звіт створений. Назва файлу — СписокВидач.docx");
        }
    }
}
```

EditCustomerForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class EditCustomerForm : Form
    {
        Customer _c;
        public string Id_V { get; set; }
        public string Id_C { get; set; }
        public string Surname_V { get; set; }
        public string Name_V { get; set; }
        public string Surname_C { get; set; }
        public string Name_C { get; set; }
        public EditCustomerForm(Customer c, string ID_v, string ID_c, string Surname_v, string Name_v,
string Surname_c, string Name_c)
        {
            Id_V = ID_v;
            Id_C = ID_c;
```

```
    Surname_V = Surname_v;
    Name_V = Name_v;
    Surname_C = Surname_c;
    Name_C = Name_c;
    _c = c;
    InitializeComponent();
}

private void EditCustomerForm_Load(object sender, EventArgs e)
{
    textBoxSurname.Text = _c.Surname;
    textBoxName.Text = _c.Name;
    textBoxPhone.Text = "0" + Convert.ToString(_c.Phone);
    textBoxBirthCity.Text = _c.BirthCity;
    textBoxBirthStreet.Text = _c.BirthStreet;
    textBoxBirthHouse.Text = _c.BirthHouse;
    textBoxFactCity.Text = _c.FactCity;
    textBoxGender.Text = _c.Gender;
    textBoxBirthDate.Text = Convert.ToString(_c.BirthDate.ToString("yyyy-MM-dd"));
    richTextBoxCustomerComment.Text = _c.Comment;

}

// Кнопка Оновити дані
private void RefreshCustomerButton_Click(object sender, EventArgs e)
{
    if (!IsDigitsOnly(textBoxPhone.Text))
    {
        MessageBox.Show("Перевірте номер телефону");
        return;
    }
    if (textBoxGender.Text != "Ч" || textBoxGender.Text != "Ж")
    {
        MessageBox.Show("Перевірте поле стать. Правильно (Ч) або (Ж)!!!");
        return;
    }
    if (IsDigitsOnly(textBoxSurname.Text))
    {
```

```
            MessageBox.Show("Перевірте прізвище");
            return;
        }
        if (IsDigitsOnly(textBoxName.Text))
        {
            MessageBox.Show("Перевірте ім'я");
            return;
        }
        MySQL mysql = new MySQL();
        try
        {
            mysql.OpenConnection();
        }
        catch
        {
            MessageBox.Show("Проблеми з доступом до бази даних!!");
            this.Close();
            return;
        }

        MySqlCommand command = new MySqlCommand("UPDATE customertable SET surname=@s,
name=@n, phone=@p, " +
            "birthcity=@bc, birthstreet=@bs, birthhouse=@bh, factcity=@fc, gender=@g, birthdate=@bd, "
+
            "comment=@c WHERE id = @id", mysql.GetConnection());
        command.Parameters.Add("@s", MySqlDbType.VarChar).Value = textBoxSurname.Text;
        command.Parameters.Add("@n", MySqlDbType.VarChar).Value = textBoxName.Text;
        command.Parameters.Add("@p", MySqlDbType.VarChar).Value = textBoxPhone.Text;
        command.Parameters.Add("@bc", MySqlDbType.VarChar).Value = textBoxBirthCity.Text;
        command.Parameters.Add("@bs", MySqlDbType.VarChar).Value = textBoxBirthStreet.Text;
        command.Parameters.Add("@bh", MySqlDbType.VarChar).Value = textBoxBirthHouse.Text;
        command.Parameters.Add("@fc", MySqlDbType.VarChar).Value = textBoxFactCity.Text;
        command.Parameters.Add("@g", MySqlDbType.VarChar).Value = textBoxGender.Text;
        command.Parameters.Add("@bd", MySqlDbType.DateTime).Value = textBoxBirthDate.Text;
        command.Parameters.Add("@c",                MySqlDbType.VarChar).Value                =
richTextBoxCustomerComment.Text;
        command.Parameters.Add("@id", MySqlDbType.VarChar).Value = Id_C;
        command.ExecuteNonQuery();
```

```
            mysql.CloseConnection();
            MessageBox.Show("Дані оновились");
        }
        // Повернутися
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
            ShowCustomCardForm NewForm = new ShowCustomCardForm(Id_V, Id_C, Surname_V, Name_V,
Surname_C, Name_C);
            NewForm.Show();
        }
        public bool IsDigitsOnly(string str)
        {
            foreach (char c in str)
            {
                if (c < '0' || c > '9')
                    return false;
            }
            return true;
        }
    }
}
```

EditVolunteerForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
```

```
{
    public partial class EditVolunteerForm : Form
    {
        Volunteer _v;
        public string _id { get; set; }
        public EditVolunteerForm(string id)
        {
            _id = id;
            InitializeComponent();
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            if (!IsDigitsOnly(textBoxPhone.Text))
            {
                MessageBox.Show("Перевірте номер телефону");
                return;
            }
            if (IsDigitsOnly(textBoxSurname.Text))
            {
                MessageBox.Show("Перевірте прізвище");
                return;
            }
            if (IsDigitsOnly(textBoxName.Text))
            {
                MessageBox.Show("Перевірте ім'я");
                return;
            }
            if (textBoxPassword.Text != textBoxPasswordAgain.Text)
            {
                MessageBox.Show("Паролі не співпадають");
                return;
            }
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
```

```
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return;
            }

            MySqlCommand command = new MySqlCommand("UPDATE volunteertable SET surname=@s,
name=@n, birthdate=@bd, " +
                "phone=@p, factcity=@fc, login=@l, password=@pw WHERE id = @id",
mysql.GetConnection());
            command.Parameters.Add("@s", MySqlDbType.VarChar).Value = textBoxSurname.Text;
            command.Parameters.Add("@n", MySqlDbType.VarChar).Value = textBoxName.Text;
            command.Parameters.Add("@bd", MySqlDbType.DateTime).Value = textBoxBirthDate.Text;
            command.Parameters.Add("@p", MySqlDbType.VarChar).Value = textBoxPhone.Text;
            command.Parameters.Add("@fc", MySqlDbType.VarChar).Value = textBoxCity.Text;
            command.Parameters.Add("@l", MySqlDbType.VarChar).Value = textBoxLogin.Text;
            command.Parameters.Add("@pw", MySqlDbType.VarChar).Value = textBoxPassword.Text;
            command.Parameters.Add("@id", MySqlDbType.VarChar).Value = _id;
            command.ExecuteNonQuery();
            mysql.CloseConnection();
            MessageBox.Show("Дані оновились");
            VolunteerListForm NewForm = new VolunteerListForm();
            NewForm.Show();
            this.Close();
        }

        private void EditVolunteerForm_Load(object sender, EventArgs e)
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
```

```
            return;
        }
        String request = "SELECT * FROM volunteertable WHERE id = @id";
        MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
        command.Parameters.Add("@id", MySqlDbType.VarChar).Value = _id;
        command.ExecuteNonQuery();
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                _v = new Volunteer(Convert.ToInt32(reader["id"]),
                    Convert.ToString(reader["surname"]),
                    Convert.ToString(reader["name"]),
                    Convert.ToDateTime(reader["birthdate"]),
                    Convert.ToInt32(reader["phone"]),
                    Convert.ToString(reader["factcity"]),
                    Convert.ToString(reader["login"]),
                    Convert.ToString(reader["password"]));
            }
        }
        mysql.CloseConnection();

        textBoxSurname.Text = _v.Surname;
        textBoxName.Text = _v.Name;
        textBoxBirthDate.Text = Convert.ToString(_v.BirthDate.ToString("yyyy-MM-dd"));
        textBoxPhone.Text = Convert.ToString(_v.Phone);
        textBoxCity.Text = _v.City;
        textBoxLogin.Text = _v.Login;
        textBoxPassword.Text = _v.Password;
        textBoxPasswordAgain.Text = _v.Password;

    }
    public bool IsDigitsOnly(string str)
    {
        foreach (char c in str)
        {
            if (c < '0' || c > '9')
                return false;
```

```
            }
            return true;
        }
    }
}
```

Form1.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace GraduateWork
{
    public partial class Form1 : Form
    {
        public string Id_C { get; set; }
        public string Id_V { get; set; }
        public string Surname_V { get; set; }
        public string Name_V { get; set; }
        public string Surname_C { get; set; }
        public string Name_C { get; set; }
        public Form1()
        {
            InitializeComponent();
        }
        public Form1(string id_volunteer, string surname, string name)
        {
            InitializeComponent();
```

```
            Id_V = id_volunteer;
            Surname_V = surname;
            Name_V = name;
        }
        // Функція очистки полів
        public void ClearTextBox()
        {
            textBoxSearchFirstName.Text = "";
            textBoxSearchLastName.Text = "";
            textBoxSearchPhoneNumber.Text = "";
        }
        // Кнопка Очистити поля пошуку
        private void ButtonClearLabel_Click(object sender, EventArgs e)
        {
            ClearTextBox();
        }

        // Кнопка Пошуку користувача
        private void ButtonSearchLabel_Click(object sender, EventArgs e)
        {
            dataGridViewShowCustomers.Rows.Clear();
            List<Customer> cust = GetCustomers();
            if(cust.Count == 0)
            {
                MessageBox.Show("Клієнта не знайдено");
                ClearTextBox();
            }
            else
            {
                foreach (Customer cu in cust)
                {
                    dataGridViewShowCustomers.Rows.Add(cu.Id, cu.Surname, cu.Name, "0"+cu.Phone,
cu.BirthCity, cu.FactCity);
                }
                foreach (DataGridViewRow r in dataGridViewShowCustomers.Rows)
                {
                    DataGridViewButtonCell ButtonCell = (DataGridViewButtonCell)r.Cells[6];
                    ButtonCell.Value = "Відкрити";
```

```
            }
            dataGridViewShowCustomers.ReadOnly = true;
        }
        labelAmountCustomer.Text = "Кількість знайдених користувачів: " +
dataGridViewShowCustomers.Rows.Count;
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        linkLabelVolunteer.Text = Surname_V + " " + Name_V;
    }

    //Отримання інфо з БД про клієнта
    public List<Customer> GetCustomers()
    {
        MySQL mysql = new MySQL();
        try
        {
            mysql.OpenConnection();
        }
        catch
        {
            MessageBox.Show("Проблеми з доступом до бази даних!!");
            this.Close();
            return null;
        }

        String sur = textBoxSearchFirstName.Text;
        String nam = textBoxSearchLastName.Text;
        String pho = textBoxSearchPhoneNumber.Text;
        String request = "SELECT * FROM customertable";
        if (sur == "" && nam == "" && pho == "")
        {
            MessageBox.Show("Не введені дані для пошуку!");
        }
        else if(sur == "" && nam == "" && pho != "")
        {
            request = "SELECT * FROM customertable WHERE phone = @p";
        }
```

```
            else if (sur == "" && nam != "" && pho != "")
            {
               request = "SELECT * FROM customertable WHERE name = @n AND phone = @p";
            }
            else if (sur != "" && nam != "" && pho != "")
            {
               request = "SELECT * FROM customertable WHERE surname = @s AND name = @n AND phone
= @p";
            }
            else if (sur != "" && nam == "" && pho != "")
            {
               request = "SELECT * FROM customertable WHERE surname = @s AND phone = @p";
            }
            else if (sur != "" && nam == "" && pho == "")
            {
               request = "SELECT * FROM customertable WHERE surname = @s";
            }
            else if (sur == "" && nam != "" && pho == "")
            {
               request = "SELECT * FROM customertable WHERE name = @n";
            }
            else if (sur != "" && nam != "" && pho == "")
            {
               request = "SELECT * FROM customertable WHERE surname = @s AND name = @n";
            }

            MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
            command.Parameters.AddWithValue("@s", sur);
            command.Parameters.AddWithValue("@n", nam);
            command.Parameters.AddWithValue("@p", pho);
            command.ExecuteNonQuery();
            List<Customer> c = new List<Customer>();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
               while (reader.Read())
               {
                  c.Add(new Customer(Convert.ToInt32(reader["id"]),
                     Convert.ToString(reader["surname"]),
```

```
                    Convert.ToString(reader["name"]),
                    Convert.ToInt32(reader["phone"]),
                    Convert.ToString(reader["birthcity"]),
                    Convert.ToString(reader["factcity"])));
                }
            }
            mysql.CloseConnection();
            return c;
        }
        // Кнопка відкриття карти клієнта
        private void dataGridViewShowCustomers_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.ColumnIndex == 6)
            {
                MySQL mysql = new MySQL();
                try
                {
                    mysql.OpenConnection();
                }
                catch
                {
                    MessageBox.Show("Проблеми з доступом до бази даних!!");
                    this.Close();
                    return;
                }
                int u = e.RowIndex;
                MessageBox.Show(dataGridViewShowCustomers.Rows[u].Cells[0].Value.ToString());
                Id_C = dataGridViewShowCustomers.Rows[u].Cells[0].Value.ToString();
                Surname_C = dataGridViewShowCustomers.Rows[u].Cells[1].Value.ToString();
                Name_C = dataGridViewShowCustomers.Rows[u].Cells[2].Value.ToString();
                this.Close();
                ShowCustomCardForm NewForm = new ShowCustomCardForm(Id_V, Id_C, Surname_V,
Name_V, Surname_C, Name_C);
                NewForm.Show();
            }
        }
        // Інфо про користувача
```

```
        private void linkLabelVolunteer_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            VolunteerCardForm NewForm = new VolunteerCardForm(Id_V, Surname_V, Name_V);
            NewForm.Show();
        }
        // Додати користувача
        private void ButtonAddCustomer_Click(object sender, EventArgs e)
        {
            AddCustomerForm NewForm = new AddCustomerForm();
            NewForm.Show();
        }
    }
}
```

Login.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    class Login
    {
        public string Surname { get; set; }
        public string Name { get; set; }
        public string Id { get; set; }
        public bool CheckUserLoginAndPassword(string _login, string _password)
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
```

```
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                return false;
            }

            MySqlCommand command = new MySqlCommand("SELECT * FROM volunteertable WHERE login = @uL AND password = @uP", mysql.GetConnection());
            command.Parameters.AddWithValue("@uL", _login);
            command.Parameters.AddWithValue("@uP", _password);
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                bool check;
                if (reader.HasRows)
                {
                    check = true;
                    while (reader.Read())
                    {
                        Surname = Convert.ToString(reader["surname"]);
                        Name = Convert.ToString(reader["name"]);
                        Id = Convert.ToString(reader["id"]);
                    }
                }
                else
                {
                    check = false;
                }
                mysql.CloseConnection();
                return check;
            }
        }

    }
}
```

LoginForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        // Кнопка Очистити поля пошуку
        private void buttonClear_Click(object sender, EventArgs e)
        {
            ClearTextBox();
        }

        // Функція очистки полів
        public void ClearTextBox()
        {
            textBoxLogin.Text = "";
            textBoxPassword.Text = "";
        }

        // Кнопка ОК
        private void buttonOk_Click(object sender, EventArgs e)
        {
            string login, password;
            if (textBoxLogin.Text == "" || textBoxPassword.Text == "")
```

```
            {
                MessageBox.Show("Не введено дані!");
                return;
            }
            login = textBoxLogin.Text;
            password = textBoxPassword.Text;
            Login autorization = new Login();
            if (autorization.CheckUserLoginAndPassword(login, password))
            {

                MessageBox.Show("Вітаю! Співробітник " + autorization.Surname + " " + autorization.Name +
" ввійшов в систему!");
                //  this.Close();
                Form1 NewForm = new Form1(autorization.Id, autorization.Surname, autorization.Name);
                NewForm.Show();
            }
            else
            {
                MessageBox.Show("Неправильний логін або пароль!!!");
                ClearTextBox();
                return;
            }

        }
    }
}
```

MovePackageForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class MovePackageForm : Form
    {
        public string id { get; set; }
        public MovePackageForm(string _id)
        {
            id = _id;
            InitializeComponent();
        }

        private void MovePackageForm_Load(object sender, EventArgs e)
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return;
            }

            MySqlCommand command = new MySqlCommand("SELECT * FROM packagetable WHERE id =
@id", mysql.GetConnection());
            command.Parameters.Add("@id", MySqlDbType.VarChar).Value = id;
            command.ExecuteNonQuery();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    labelname.Text = Convert.ToString(reader["name"]);
                    textBox1.Text = Convert.ToString(reader["amount"]);
```

```
                labelcomment.Text = Convert.ToString(reader["comment"]);
            }
        }
        mysql.CloseConnection();


    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (!IsDigitsOnly(textBox1.Text))
        {
            MessageBox.Show("Перевірте кількість пакунків");
            return;
        }
        MySQL mysql = new MySQL();
        try
        {
            mysql.OpenConnection();
        }
        catch
        {
            MessageBox.Show("Проблеми з доступом до бази даних!!");
            this.Close();
            return;
        }

        MySqlCommand command = new MySqlCommand("UPDATE packagetable SET amount = @amount WHERE id = @id", mysql.GetConnection());
        command.Parameters.Add("@amount", MySqlDbType.VarChar).Value = textBox1.Text;
        command.Parameters.Add("@id", MySqlDbType.VarChar).Value = id;
        command.ExecuteNonQuery();
        mysql.CloseConnection();
        MessageBox.Show("Дані оновились");
    }

    private void button2_Click(object sender, EventArgs e)
    {
```

```
            this.Close();
        }
        public bool IsDigitsOnly(string str)
        {
            foreach (char c in str)
            {
                if (c < '0' || c > '9')
                    return false;
            }
            return true;
        }
    }
}
```

MySQL.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace GraduateWork
{
    class MySQL
    {
        MySqlConnection connection = new MySqlConnection("server = localhost; port = 3306; " +
            "username = root; password = ''; database = volunteerorganization");
        public void OpenConnection()
        {
            if (connection.State == System.Data.ConnectionState.Closed)
            {
                connection.Open();
            }
```

```
        }
        public void CloseConnection()
        {
            if (connection.State == System.Data.ConnectionState.Open)
            {
                connection.Close();
            }
        }
        public MySqlConnection GetConnection()
        {
            return connection;
        }
    }
}
```

Package.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraduateWork
{
    public class Package
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public int Amount { get; set; }
        public string Comment { get; set; }

        public Package(string id, string _n, int _a, string _c)
        {
            Id = id;
            Name = _n;
            Amount = _a;
```

```
            Comment = _c;
        }

    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new AdminPanel());
        }
    }
}
```

ShowDeliveryForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class ShowAllDeliveryForm : Form
    {
        public ShowAllDeliveryForm()
        {
            InitializeComponent();
            AllPackageLoad();
        }

        public void AllPackageLoad()
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return;
            }

            String request = "SELECT * FROM packagetable";

            MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
            command.ExecuteNonQuery();
            List<Package> p = new List<Package>();
            using (MySqlDataReader reader = command.ExecuteReader())
```

```
            {
                while (reader.Read())
                {
                    p.Add(new Package(Convert.ToString(reader["id"]),
                        Convert.ToString(reader["name"]),
                        Convert.ToInt32(reader["amount"]),
                        Convert.ToString(reader["comment"])));
                }
            }
            int _a = 0;
            foreach (Package _p in p)
            {
                PackagedataGridView.Rows.Add(_p.Name, _p.Amount, _p.Comment);
                _a += _p.Amount;
            }
            PackagedataGridView.ReadOnly = true;
            labelAmountAllPackage.Text = "Кількість всіх доступних пакунків: " + _a;
            labelAmountKindPackages.Text = "Кількість видів пакунків: " +
PackagedataGridView.Rows.Count;
        }
    }
}
```

ShowCustomCardForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class ShowCustomCardForm : Form
    {
        public string ID_C { get; set; } // id користувача
        public string ID_V { get; set; } // id волонтера
        public string ID_D { get; set; } // id видачі
```

```
        public string S_V { get; set; } // прізвище волонтера
        public string N_V { get; set; } // ім'я волонтера
        public string S_C { get; set; } // прізвище користувача
        public string N_C { get; set; } // ім'я користувача

        public ShowCustomCardForm(string id_v, string id_c, string surname_v, string name_v, string surname_c, string name_c)
        {
            InitializeComponent();
            ID_C = id_c;
            ID_V = id_v;
            S_V = surname_v;
            N_V = name_v;
            S_C = surname_c;
            N_C = name_c;
        }

        public ShowCustomCardForm()
        {
            InitializeComponent();
        }

        // Загрузка карти клієнта
        private void ShowCustomCard_Load(object sender, EventArgs e)
        {
            ShowCustomerInfo();
            ShowDataGridDelivery();
            labelAmountDelivery.Text = "Кількість видач в користувача: " + ShowAllDeliverydataGridView.Rows.Count;

        }
        // Завантаження інфо про видачі в датагрід
        public void ShowDataGridDelivery()
        {
            ShowAllDeliverydataGridView.Rows.Clear();
            List<Delivery> deliv = GetDeliveries();
            if (deliv.Count == 0)
            {
```

```
                MessageBox.Show("Видач у клієнта не знайдено");
            }
            else
            {
                foreach (Delivery d in deliv)
                {
                    ShowAllDeliverydataGridView.Rows.Add(d.Id, d.Date.ToString("dd/MM/yyyy"),
d.Name_Package, d.FullName_Volunteer);
                }
                foreach (DataGridViewRow r in ShowAllDeliverydataGridView.Rows)
                {
                    DataGridViewButtonCell ButtonCell = (DataGridViewButtonCell)r.Cells[4];
                    ButtonCell.Value = "Видалити";
                }
                ShowAllDeliverydataGridView.ReadOnly = true;

            }
        }
        // Завантаження інфо про користувача
        public void ShowCustomerInfo()
        {
            List<Customer> cust = GetCustomer(ID_C);
            MessageBox.Show(cust[0].Surname.ToString());
            labelSurnameShow.Text = cust[0].Surname.ToString();
            labelNameShow.Text = cust[0].Name.ToString();
            labelPhoneShow.Text = "0" + cust[0].Phone.ToString();
            labelBirthCityShow.Text = cust[0].BirthCity.ToString();
            labelBirthStreetShow.Text = cust[0].BirthStreet.ToString();
            labelBirthHouseShow.Text = cust[0].BirthHouse.ToString();
            labelFactCityShow.Text = cust[0].FactCity.ToString();
            labelGenderShow.Text = cust[0].Gender.ToString();
            labelBirthDateShow.Text = cust[0].BirthDate.ToString("yyyy-MM-dd"); //dd MMMM yyyy
            richTextBoxCustomerComment.Text = cust[0].Comment.ToString();
        }
        // Отримання інфо з БД про клієнта
        public List<Customer> GetCustomer(string id)
        {
            MySQL mysql = new MySQL();
```

```
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return null;
            }

            MySqlCommand command = new MySqlCommand("SELECT * FROM customertable WHERE id
= @id", mysql.GetConnection());
            command.Parameters.AddWithValue("@id", id);
            command.ExecuteNonQuery();
            List<Customer> c = new List<Customer>();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    c.Add(new Customer(Convert.ToString(reader["surname"]),
                        Convert.ToString(reader["name"]),
                        Convert.ToInt32(reader["phone"]),
                        Convert.ToString(reader["birthcity"]),
                        Convert.ToString(reader["birthstreet"]),
                        Convert.ToString(reader["birthhouse"]),
                        Convert.ToString(reader["factcity"]),
                        Convert.ToString(reader["gender"]),
                        Convert.ToDateTime(reader["birthdate"]),
                        Convert.ToString(reader["comment"])));
                }
            }
            mysql.CloseConnection();
            return c;
        }
        // Отримання інфо з БД про видачі
        public List<Delivery> GetDeliveries()
        {
```

```
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return null;
            }

            String request = "SELECT dt.id AS 'del_id', dt.date AS 'del_date', pt.name AS 'pac_name', vt.surname AS 'vol_surname', vt.name AS 'vol_name' FROM deliverytable dt JOIN packagetable pt ON dt.packageid = pt.id JOIN volunteertable vt ON dt.volunteerid = vt.id WHERE customerid = @C_id";

            MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
            command.Parameters.AddWithValue("@C_id", ID_C);
            command.ExecuteNonQuery();
            List<Delivery> c = new List<Delivery>();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    c.Add(new Delivery(Convert.ToInt32(reader["del_id"]),
                        Convert.ToDateTime(reader["del_date"]),
                        Convert.ToString(reader["pac_name"]),
                        Convert.ToString(reader["vol_surname"]) + " " + reader["vol_name"]));
                }
            }
            mysql.CloseConnection();
            return c;
        }
        // Кнопка Додати видачу
        private void AddDeliveryButton_Click(object sender, EventArgs e)
        {
            this.Close();
            AddDeliveryForm NewForm = new AddDeliveryForm(ID_V, ID_C, S_V, N_V, S_C, N_C);
```

```
            NewForm.Show();
        }
        // Кнопки видалення/редагування видач
        private void ShowAllDeliverydataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.ColumnIndex == 4)
            {
                // Кнопка видалення видачі
                MySQL mysql = new MySQL();
                try
                {
                    mysql.OpenConnection();
                }
                catch
                {
                    MessageBox.Show("Проблеми з доступом до бази даних!!");
                    this.Close();
                    return;
                }
                int u = e.RowIndex;
                ID_D = ShowAllDeliverydataGridView.Rows[u].Cells[0].Value.ToString();
                DialogResult dialogResult = MessageBox.Show("Ви хочете видалити цю видачу?",
"Підтвердження видалення", MessageBoxButtons.YesNo);
                if (dialogResult == DialogResult.Yes)
                {
                    MySqlCommand command = new MySqlCommand("DELETE FROM deliverytable WHERE
id = @id", mysql.GetConnection());
                    command.Parameters.AddWithValue("@id",
ShowAllDeliverydataGridView.Rows[e.RowIndex].Cells[0].Value);
                    command.ExecuteNonQuery();
                    ShowAllDeliverydataGridView.Rows.RemoveAt(e.RowIndex);
                    MessageBox.Show("Успішно видалено видачу");
                }
                labelAmountDelivery.Text = "Кількість видач в користувача: " +
ShowAllDeliverydataGridView.Rows.Count;
            }
        }
```

```
        // Кнопка оновити датагрід після додавання видачі
        private void RefreshTableButton_Click(object sender, EventArgs e)
        {
            ShowCustomerInfo();
            ShowDataGridDelivery();
            labelAmountDelivery.Text = "Кількість видач в користувача: " + ShowAllDeliverydataGridView.Rows.Count;
            MessageBox.Show("Оновлено інформацію по користувачу");
        }

        // Видалити дані про користувача
        private void DeleteCustomerButton_Click(object sender, EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Ви хочете видалити всі дані про цього користувача?", "Підтвердження видалення", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                MySQL mysql = new MySQL();
                try
                {
                    mysql.OpenConnection();
                }
                catch
                {
                    MessageBox.Show("Проблеми з доступом до бази даних!!");
                    this.Close();
                    return;
                }

                MySqlCommand command = new MySqlCommand("DELETE FROM deliverytable WHERE customerid = @id", mysql.GetConnection());
                command.Parameters.AddWithValue("@id", ID_C);
                command.ExecuteNonQuery();
                MessageBox.Show("Видалив з делівері");
                command = new MySqlCommand("DELETE FROM customertable WHERE id = @id", mysql.GetConnection());
                command.Parameters.AddWithValue("@id", ID_C);
```

```
                command.ExecuteNonQuery();

                MessageBox.Show("Видалив з кастомер");
                mysql.CloseConnection();

                //    this.Close();
            }
        }

        // Редагувати дані про користувача
        private void EditCustomerButton_Click(object sender, EventArgs e)
        {
            Customer c = new Customer(Convert.ToString(labelSurnameShow.Text),
Convert.ToString(labelNameShow.Text),
                Convert.ToInt32(labelPhoneShow.Text), Convert.ToString(labelBirthCityShow.Text),
                Convert.ToString(labelBirthStreetShow.Text), Convert.ToString(labelBirthHouseShow.Text),
                Convert.ToString(labelFactCityShow.Text), Convert.ToString(labelGenderShow.Text),
                Convert.ToDateTime(labelBirthDateShow.Text),
Convert.ToString(richTextBoxCustomerComment.Text));
            this.Close();
            EditCustomerForm NewForm = new EditCustomerForm(c, ID_V, ID_C, S_V, N_V, S_C, N_C);
            NewForm.Show();
        }

        private void ShowAllDeliveryButton_Click(object sender, EventArgs e)
        {
            ShowAllDeliveryForm NewForm = new ShowAllDeliveryForm();
            NewForm.Show();
        }
        // Повернутися
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
            Form1 NewForm = new Form1(ID_V, S_V, N_V);
            NewForm.Show();
        }
    }
}
```

Volunteer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraduateWork
{
   public class Volunteer
   {
      public int Id { get; set; }
      public string Surname { get; set; }
      public string Name { get; set; }
      public DateTime BirthDate { get; set; }
      public int Phone { get; set; }
      public string City { get; set; }
      public string Login { get; set; }
      public string Password { get; set; }
      public string AmountDelivery { get; set; }

      public Volunteer(int id, string surname, string name, DateTime birthdate, int phone, string city, string login, string password, string volunteerid, string count)
      {
         Id = id;
         Surname = surname;
         Name = name;
         BirthDate = birthdate;
         Phone = phone;
         City = city;
         Login = login;
         Password = password;
         if(volunteerid == "")
         {
            AmountDelivery = "0";
         }
```

```
            else
            {
                AmountDelivery = count;
            }
        }
        public Volunteer(int id, string surname, string name, DateTime birthdate, int phone, string city, string login, string password)
        {
            Id = Id;
            Surname = surname;
            Name = name;
            BirthDate = birthdate;
            Phone = phone;
            City = city;
            Login = login;
            Password = password;
        }
    }
}
```

VolunteerCardForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraduateWork
{
    public partial class VolunteerCardForm : Form
    {
```

```
public string Id_V { get; set; }
public string Sur_V { get; set; }
public string Nam_V { get; set; }
public VolunteerCardForm()
{
    InitializeComponent();
}
public VolunteerCardForm(string id, string surname, string name)
{
    Id_V = id;
    Sur_V = surname;
    Nam_V = name;
    InitializeComponent();
}

private void VolunteerCardForm_Load(object sender, EventArgs e)
{
    MySQL mysql = new MySQL();
    try
    {
        mysql.OpenConnection();
    }
    catch
    {
        MessageBox.Show("Проблеми з доступом до бази даних!!");
        this.Close();
        return;
    }
    String request = "SELECT * FROM volunteertable WHERE id = @id";
    MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
    command.Parameters.AddWithValue("@id", Id_V);
    command.ExecuteNonQuery();
    List<Customer> c = new List<Customer>();
    using (MySqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            labelSurnameShow.Text = Convert.ToString(reader["surname"]);
```

```
                    labelNameShow.Text = Convert.ToString(reader["name"]);
                    labelPhoneShow.Text = "0" + Convert.ToString(reader["phone"]);
                    labelBirthCityShow.Text = Convert.ToString(reader["factcity"]);
                    labelBirthDateShow.Text =
Convert.ToDateTime(reader["birthdate"]).ToString("dd/MM/yyyy");
                }
            }
            mysql.CloseConnection();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();

        }
    }
}
```

VolunteerListForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Xceed.Document.NET;
using Xceed.Words.NET;

namespace GraduateWork
{
    public partial class VolunteerListForm : Form
    {
```

```
        public VolunteerListForm()
        {
            InitializeComponent();
        }

        private void VolunteerListForm_Load(object sender, EventArgs e)
        {
            List<Volunteer> _vol = GetVolunteers();
            if (_vol.Count == 0)
            {
                MessageBox.Show("Клієнта не знайдено");
                //  ClearTextBox();
            }
            else
            {
                foreach (Volunteer v in _vol)
                {
                    dataGridViewVolunteerList.Rows.Add(v.Id, v.Surname, v.Name,
v.BirthDate.ToString("dd/MM/yyyy"), "0" + v.Phone, v.City, v.AmountDelivery);
                }
                foreach (DataGridViewRow r in dataGridViewVolunteerList.Rows)
                {
                    DataGridViewButtonCell ButtonCell = (DataGridViewButtonCell)r.Cells[7];
                    ButtonCell.Value = "Редагувати";

                    ButtonCell = (DataGridViewButtonCell)r.Cells[8];
                    ButtonCell.Value = "Видалити";

                }
                dataGridViewVolunteerList.ReadOnly = true;
            }
            labelAmountVolunteer.Text = "Кількість волонтерів: " + dataGridViewVolunteerList.Rows.Count;
        }

        public List<Volunteer> GetVolunteers()
        {
            MySQL mysql = new MySQL();
            try
```

```
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return null;
            }
            String request = "SELECT vt.id, vt.surname, vt.name, vt.birthdate, vt.phone, vt.factcity, " +
                "vt.login, vt.password,  dt.volunteerid, COUNT(*) AS count_del " +
                "FROM volunteertable vt LEFT JOIN deliverytable dt ON vt.id = dt.volunteerid GROUP BY
vt.id";
            MySqlCommand command = new MySqlCommand(request, mysql.GetConnection());
            command.ExecuteNonQuery();
            List<Volunteer> v = new List<Volunteer>();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    v.Add(new Volunteer(Convert.ToInt32(reader["id"]),
                        Convert.ToString(reader["surname"]),
                        Convert.ToString(reader["name"]),
                        Convert.ToDateTime(reader["birthdate"]),
                        Convert.ToInt32(reader["phone"]),
                        Convert.ToString(reader["factcity"]),
                        Convert.ToString(reader["login"]),
                        Convert.ToString(reader["password"]),
                        Convert.ToString(reader["volunteerid"]),
                        Convert.ToString(reader["count_del"])));
                }
            }
            mysql.CloseConnection();
            return v;
        }

        private void buttonAddVolunteer_Click(object sender, EventArgs e)
        {
```

```
            AddVolunteerForm NewForm = new AddVolunteerForm();
            NewForm.Show();
            this.Close();
        }
        // Формування звіту
        private void button1_Click(object sender, EventArgs e)
        {
            string WordDocumentPath = "СписокВолонтерів.docx";
            DocX document = DocX.Create(WordDocumentPath);

            Paragraph paragraph1 = document.InsertParagraph();
            paragraph1.AppendLine("Кількість Волонтерів " + dataGridViewVolunteerList.Rows.Count)
                .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                .FontSize(16);
            paragraph1.AppendLine("Список Волонтерів: ")
                .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                .FontSize(16);
            Table t = document.AddTable(dataGridViewVolunteerList.Rows.Count + 1,
dataGridViewVolunteerList.Columns.Count);
            t.Alignment = Alignment.left;
            t.Design = TableDesign.ColorfulListAccent5;

            for (int i = 0; i < 6; i++)
            {
                t.Rows[0].Cells[i].Paragraphs[0].Append(dataGridViewVolunteerList.Columns[i].HeaderText)
                    .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
                    .FontSize(14)
                    .Bold();
            }
            for (int i = 1; i < t.Rows.Count; i++)
            {
                for (int j = 0; j < 6; j++)
                {
                    object SomeVal = dataGridViewVolunteerList[j, i - 1].Value;
                    if (SomeVal != null)
                    {
                        t.Rows[i].Cells[j].Paragraphs[0].Append(SomeVal.ToString())
                            .Font(new Xceed.Document.NET.Font("TimesNewRoman"))
```

```
                    .FontSize(10);
                }
            }
        }
        document.InsertTable(t);

        Paragraph paragraph2 = document.InsertParagraph();
        document.Save();
        MessageBox.Show("Звіт створений. Назва файлу — СписокВолонтерів.docx");

    }

    private void dataGridViewVolunteerList_CellContentClick(object sender, DataGridViewCellEventArgs
e)
    {
        if (e.ColumnIndex == 7)
        {
            MySQL mysql = new MySQL();
            try
            {
                mysql.OpenConnection();
            }
            catch
            {
                MessageBox.Show("Проблеми з доступом до бази даних!!");
                this.Close();
                return;
            }
            int u = e.RowIndex;
            string id = dataGridViewVolunteerList.Rows[u].Cells[0].Value.ToString();
            EditVolunteerForm NewForm = new EditVolunteerForm(id);
            NewForm.Show();
            this.Close();
        }
        if (e.ColumnIndex == 8)
        {
            int u = e.RowIndex;
            if (Convert.ToInt32(dataGridViewVolunteerList.Rows[u].Cells[6].Value) > 0)
```

```
                {
                    DialogResult dialogResult = MessageBox.Show("Волонтера не можна видалити, адже в нього є видачі.\n\nДодати приставку до його прізвище «звіл_»?", "Message", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        MySQL mysql = new MySQL();
                        try
                        {
                            mysql.OpenConnection();
                        }
                        catch
                        {
                            MessageBox.Show("Проблеми з доступом до бази даних!!");
                            this.Close();
                            return;
                        }

                        MySqlCommand command = new MySqlCommand("UPDATE volunteertable SET surname=@s WHERE id = @id", mysql.GetConnection());
                        command.Parameters.Add("@s", MySqlDbType.VarChar).Value = "звіл_" + dataGridViewVolunteerList.Rows[u].Cells[1].Value.ToString();
                        command.Parameters.Add("@id", MySqlDbType.VarChar).Value = dataGridViewVolunteerList.Rows[u].Cells[0].Value.ToString();
                        command.ExecuteNonQuery();
                        mysql.CloseConnection();
                        MessageBox.Show("Дані оновились");
                        VolunteerListForm NewForm = new VolunteerListForm();
                        NewForm.Show();
                        this.Close();
                    }
                }
                else
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що хочете видалити волонтера " + dataGridViewVolunteerList.Rows[u].Cells[1].Value + " " + dataGridViewVolunteerList.Rows[u].Cells[2].Value + " ?", "Message", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
```

```
                    MySQL mysql = new MySQL();
                    try
                    {
                       mysql.OpenConnection();
                    }
                    catch
                    {
                       MessageBox.Show("Проблеми з доступом до бази даних!!");
                       this.Close();
                       return;
                    }

                    MySqlCommand command = new MySqlCommand("DELETE FROM volunteertable WHERE id = @id", mysql.GetConnection());
                    command.Parameters.Add("@id", MySqlDbType.VarChar).Value = dataGridViewVolunteerList.Rows[u].Cells[0].Value.ToString();
                    command.ExecuteNonQuery();
                    mysql.CloseConnection();
                    MessageBox.Show("Користувача видалено");
                    VolunteerListForm NewForm = new VolunteerListForm();
                    NewForm.Show();
                    this.Close();
                  }
               }
            }
         }
      }
```