МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Циклова комісія програмної інженерії

К У Р С О В И Й П Р О Е К Т

з навчальної дисципліни
"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

на тему: Програмний модуль «Фоторепортер» обліку замовлень
(вказати тему курсового проекту)

Студента IV курсу ПЗ-19-1 групи
напряму підготовки 6.050103
Програмна Інженерія
спеціальності 121 Інженерія
програмного забезпечення
Вакуленко О.Д.
(прізвище та ініціали студента)
Керівник викладач Гапоненко Н.В.
Національна шкала_____
Кількість балів: _____ Оцінка ECTS:_____

Члени комісії _____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)

м. ДНІПРО
2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Предметна комісія програмної інженерії

ЗАТВЕРДЖУЮ
Голова предметної комісії ПІ
_____
С.С.Ланська
"__"_____ 2022 р.

ЗАВДАННЯ
на виконання курсового проєкту

з дисципліни Об'єктно-орієнтоване програмування

студенту _____
(прізвище, ім'я та по батькові)

Відділення Комп'ютерної та програмної інженерії

Спеціальність 121 Інженерія програмного забезпечення

Курс IV Група (шифр) ПЗ-19-1

1 Тема проекту Програмний модуль «Фоторепортер» обліку замовлень фотосесій

2 Початкові дані: _____

Розглянуто і ухвалено на засіданні циклової комісії Програмної інженерії
Протокол № 2 від 16.09.2020 р.

Керівник КП _____ Н.В.Гапоненко
(підпис) (ініціали та прізвище)

Завдання до виконання
одержав студент _____ _____
(підпис) (ініціали та прізвище)

Дата видачі

Термін виконання

Зміст

Вступ

У сучасному світі повсюдно необхідна автоматизація належного зберігання та обліку цих даних. Майже кожна компанія має власне програмне забезпечення, яке допомагає співробітникам виконувати свою роботу.

Студії мають багато програмного забезпечення, але ці програми важко освоїти та використовувати. Велика частина цих програм працює в Інтернеті, що не завжди зручно.

Програма має бути простою у використанні, швидкою, легкою для вивчення та придатною для багатьох малих і великих підприємств. Користуватися програмою можуть навіть недосвідчені користувачі ПК. Його може використовувати як одна людина (наприклад, керівник невеликої компанії), так і група людей (наприклад, у великій компанії робота розбита на кілька посад).

1 ПОСТАНОВКА ЗАДАЧІ

Необхідно розробити програмний продукт який дозволить заказати послуги різного напрямку. Забронювати послугу на конкретний час а отримати приблизний термін виконання.

Наприкінці отримати відповідну форму з інформацією про послугу термін, час виконання та ціну.

Детальний опис функціонала:

- Можливість додавання замовлень
- Можливість зміненя замовлень
- Можливість додавання/зміненя послуг

Вхідні дані:

1 Дані замовника:
   - ID замовника
   - ФІО замовника
   - Номер телефону замовника

2 Дані Послуги:
   - ID послуги
   - Назва послуги
   - Тип послуги
   - Додаток до послуги

3 Дані замовлення:
   - ID замовлення
   - Дата замовлення
   - Дата виконання
   - Посилання на малюнок

- Адреса замовлення
- Ціна замовлення
- Статус замовлення
- ID клієнта
- ID послуги

Вимоги до пристрою:

- Операційна система: Windows 10
- СУБД: MySQL, ODBC

2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1 Опис середовища програмування

Qt — це кросплатформний набір для розробки програмного забезпечення для мови програмування C++. Дозволяє запускати програмне забезпечення, написане на ньому, у більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної ОС, не змінюючи вихідний код. Містить усі базові класи, які можуть знадобитися для розробки прикладного програмного забезпечення, від елементів графічного інтерфейсу до мереж, баз даних, класів OpenGL, SVG і XML. Бібліотека забезпечує потоковий, мережевий і міжплатформний доступ до файлів. Qt також доступний для багатьох інших мов програмування: Ada (QtAda), C# (Qyoto/Kimono), Java (Qt Jambi), Qt Jambi, Node.js, Pascal, Perl, PHP (PHP-Qt), Ruby (QtRuby) і Python (PyQt, PySide).

Qt 5 був випущений у грудні 2012 року з його модульною структурою та перемістіть увагу на написання програм за допомогою інструментів декларативний опис інтерфейсу, який визначає логіку взаємодії з користувачем н мові JavaScript, тоді як програми C++ позиціонуються для реалізації критичні для виконання або надто складні частини програми, а також с творіть новий модульний сервер для Qt Quick. Хоча багато основні вдосконалення та зміни, Qt 5 зберігає базову зворотну сумісність із минулим версія, яка повністю підтримує інструменти для створення програм Qt цією мовою C++ і включає майже всі компоненти Qt 4 (давно застаріли елементи), більшість модулів колишнього Qt Mobility та деякі експериментальні модулі Проект від Qt Labs.

З моменту створення в 1996 році комерційна версія бібліотеки Qt потрапила в розряд основа для тисяч успішних проектів по всьому світу.

Крім того, Qt є фундаментом популярного робочого середовища KDE, що входить до складу багатьох дистрибутивів GNU/Linux.

2.2 Опис мови програмування

Для реалізації цього проекту були обрані такі мови програмування: - C++ (діалект Qt/MOC і автоматизований QMake) - QML (бібліотека QStyle). C++ — мова програмування високого рівня, яка підтримує кілька парадигм Програмування: об'єктно-орієнтоване, узагальнене та процедурне. розроблений Б'ярне Страуструп з AT&T Bell Labs

При створенні C++ прагнули зберегти сумісність з мовою C. Більшість програм на C справно працюватимуть і з компілятором C++. C++ має синтаксис, заснований на синтаксисі C.

Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;

− посилання і оператори управління вільно розподіленою пам'яттю.

QML (Qt Meta Language або Qt Modeling Language) - декларативна мова програмування, в основі якої лежить мова JavaScript. QML використовується для розробки додатків, які роблять основний упор на призначений для користувача інтерфейс і, в цілому, на дизайн графічної частини. Є частиною Qt Quick, середовища розробки призначеного для користувача інтерфейсу, поширюваної разом з Qt. Часто використовується для створення додатків, орієнтованих на мобільні пристрої з сенсорним управлінням.

QML-документ являє собою дерево елементів. QML елемент, так само, як і елемент Qt, являє собою сукупність блоків: графічних (таких, як rectangle, image) і поведінкових (таких, як state, transition, animation). Ці елементи можуть бути об'єднані, щоб побудувати комплексні компоненти, починаючи від простих кнопок і повзунків і закінчуючи повноцінними додатками, що працюють з інтернетом.

QML елементи можуть бути доповнені стандартними для JavaScript вставками шляхом вбудовування .js файлів. Також вони можуть бути розширені C++ компонентами через Qt framework.

2.3 Опис СКБД

MySQL — безкоштовна система керування реляційними базами даних. MySQL розроблена компанією «ЦХ» для підвищення швидкості обробки великих баз даних. Ця система керування базами даних з відкритим кодом (СУБД) була створена як заміна комерційним системам. З самого початку MySQL була дуже схожа на mSQL, але з часом вона розширилася, і зараз MySQL є однією з

найпопулярніших систем керування базами даних. Він в основному використовується для створення динамічних веб-сторінок через чудову підтримку кількох мов програмування.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

2.4 Опис основних принципів ООП

Інкапсуляція

Вирішальним фактором у проектуванні компонента програми є приховування внутрішніх даних компонента та деталей його реалізації від інших компонентів програми, а також надання набору методів (API) для взаємодії з ними. Цей принцип є одним із трьох фундаментальних принципів ООП і називається інкапсуляцією.

Наслідування

Наслідування є одним із найважливіших принципів об’єктно-орієнтованого програмування, оскільки воно дозволяє створювати ієрархії об’єктів. Використовуючи успадкування, ви можете створити загальний клас

Поліморфізм

Розглядаючи поліморфізм, слід пам’ятати, що цей принцип нерозривно пов’язаний з іншим принципом об’єктно-орієнтованого програмування – імітацією, який допомагає досягти поліморфізму.

Давайте візьмемо приклад абстрактного класу «Автомобіль», за яким слідують два конкретних класи — «Спортивний автомобіль» і «Вантажівка».

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис бази даних

Для збереження великих об’ємів інформації потрібно розробити низку правил та відношень між цими даними. Відношення між таблицями показані на ER-діаграмі. Зміст таблиць бази даних 3.1 - 3.4

Таблиця 3.1 - Користувач(users)

Назва поля	Опис	Тип	Розмір	Ключ
Users_ID	Унікальний код	Числовий	Макс. значеня – 2147483647	Первинний
Users_name	Ім’я лікаря	Символьний	Макс. Кількість символів - 45	
Users_surname	Прізвище лікаря	Символьний	Макс. Кількість символів - 45	
Users_patronymic	По-батькові	Символьний	Макс. Кількість символів - 45	
Phone_number	Номер телефону	Символьний	Макс. Кількість символів - 12	

Таблиця 3.2 - Послуги(iservices)

Назва поля	Опис	Тип	Розмір	Ключ
Service_ID	Унікальний код	Числовий	Макс. значеня – 2147483647	Первинний
Service_name	Назва послуги	Символьний	Макс. Кількість	

			символів - 45	
Service_type	Тип послуги	Символьний	Макс. Кількість символів - 45	
Service_price	Ціна послуги	Числовий	Макс. значеня – 2147483647	
commentary	Коментар до послуги	Символьний	Макс. Кількість символів - 255	

Таблиця 3.3 - Данні прийому(order_data)

Назва поля	Опис	Тип	Розмір	Ключ
Order_id	Унікальний код	Числовий	Макс. значеня – 2147483647	Первинний
Creation_date	Дата створення	DATE	Стандарт ISO	Зовнішній
Fianlization_date	Дата прийому	DATE	Стандарт ISO	
img	Посилання на зображення	Символьний	Стандарт ISO	
addres	Адреса	Символьний	Стандарт ISO	

Order_price	ціна	Числовий	Макс. значеня – 2147483647	
Order_status	статус	BOOL	Макс. значеня – 1	
Users_ID	Унікальний код	Числовий	Макс. значеня – 2147483647	Зовнішній
Service_ID	Унікальний код	Числовий	Макс. значеня – 2147483647	Зовнішній

ER-діаграма зв’язку таблиць бази даних приведена нижче.

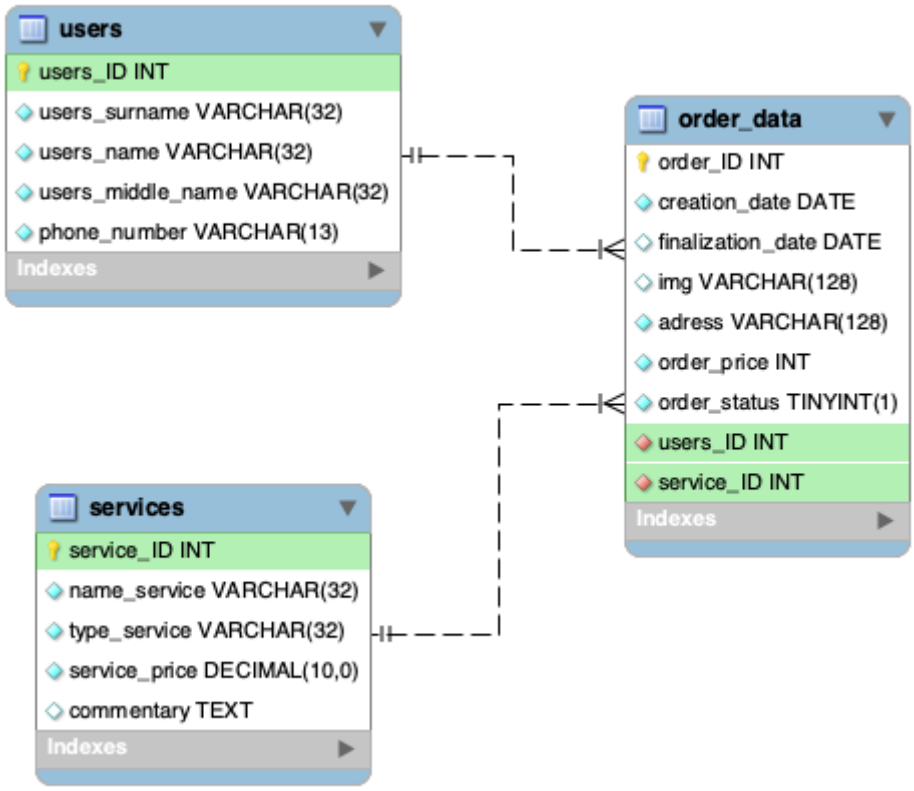

Рисунок 3.1 - ER-діаграма зв’язку таблиць бази даних

3.2 Проектування користувацького інтерфейсу

Головне вікно містить три блоки: “Замовлення”, “Облікові дані”, “Дії”. Блок “Замовлення” містить пошукове поле та таблицю, у якій виводяться дані про замовлення. Блок “Облікові дані” містить поля для вводу даних та кнопку для збереження змін у цих полях до бази даних. Блок “Дії” містить кнопку “Додати”, яка при натисканні на неї відкриває нове вікно, для створення нового замовлення, кнопку ” Оновити”, яка поновлю таблицю та кнопку “Видалити” яка видаляю обраний елемент у таблиці з замовленнями.

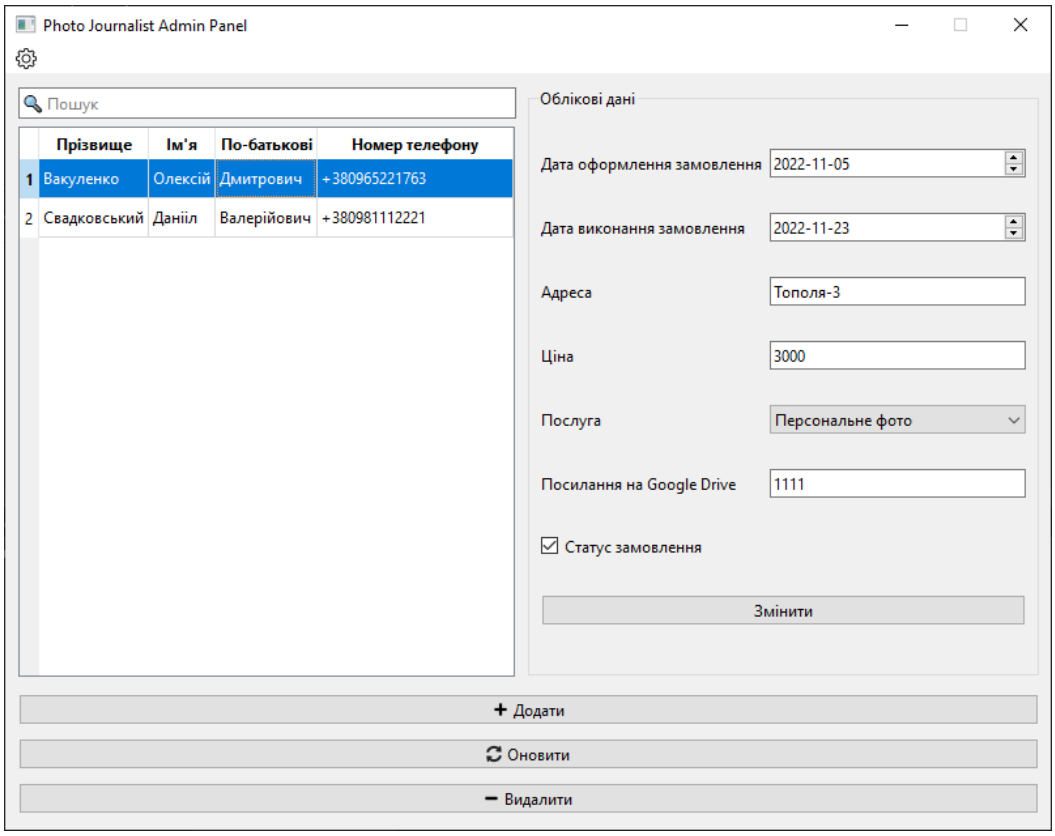

Рисунок 3.2 - Головне вікно програми

Вікно додавання нового замовлення містить два блоки.

У першому блоці розташовані поля для введення даних клієнта.

У другому блоці розміщується поля для детальної інформації про замовлення, тобто кінець виконання замовлення, адреса, за якою замовлення буде виконане, список із послугами та поле посилання на гугл диск у якому розташоване фото. Також це поле містить прапорець та кнопку. Якщо нажати на прапорець то поля кінцевої дати та посилання на гугл диск будуть активними. Якщо нажати на кнопку “Далі” то введені дані додадуться до бази даних та вікно закриється та відкриється минуле, у протилежному випадку повернеться вікно за попередженням про помилку.

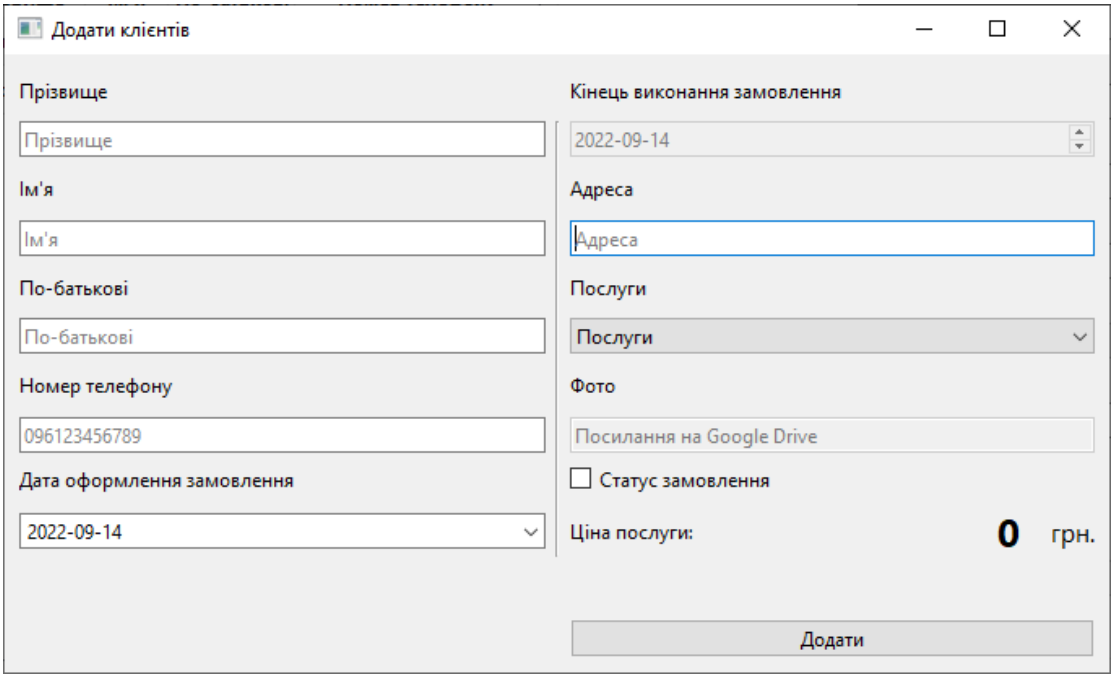

Рисунок 3.3 - Вікно додавання замовлення

Вікно додавання послуги містить поля для вводу даних про послугу, такі як:

1. Назва
2. Вид
3. Ціна
4. Примітка або додаток

Також це вікно має кнопку “Додати”, яке додає нові значення до бази даних видає інформаційне вікно про вдале додавання нового елементу та повертається до минулого вікна. У зворотному випадку повертає інформаційне вікно про помилку.

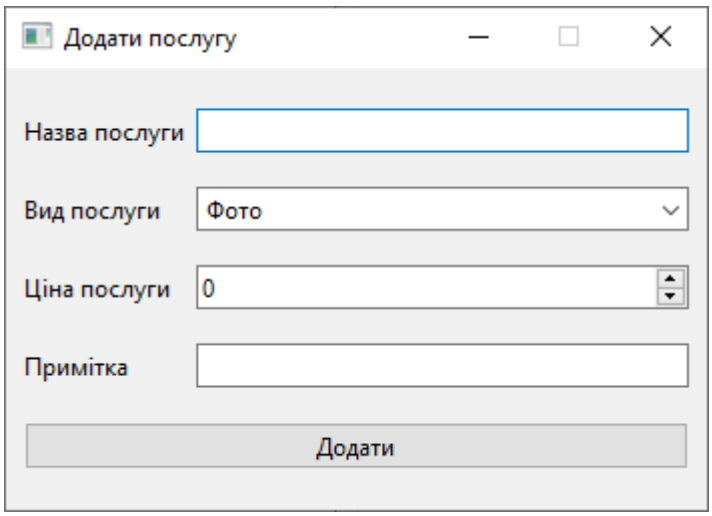

Рисунок 3.5 - Додавання послуги

Вікно видалення послуги містить таблицю із усіма можливими послугами у базі даних та кнопкою “Видалити” яка стає активною

якщо обрати хоча б один елемент із таблиці. Та при натисканні видаляє цей об’єкт за бази даних та оновлює дану таблицю.

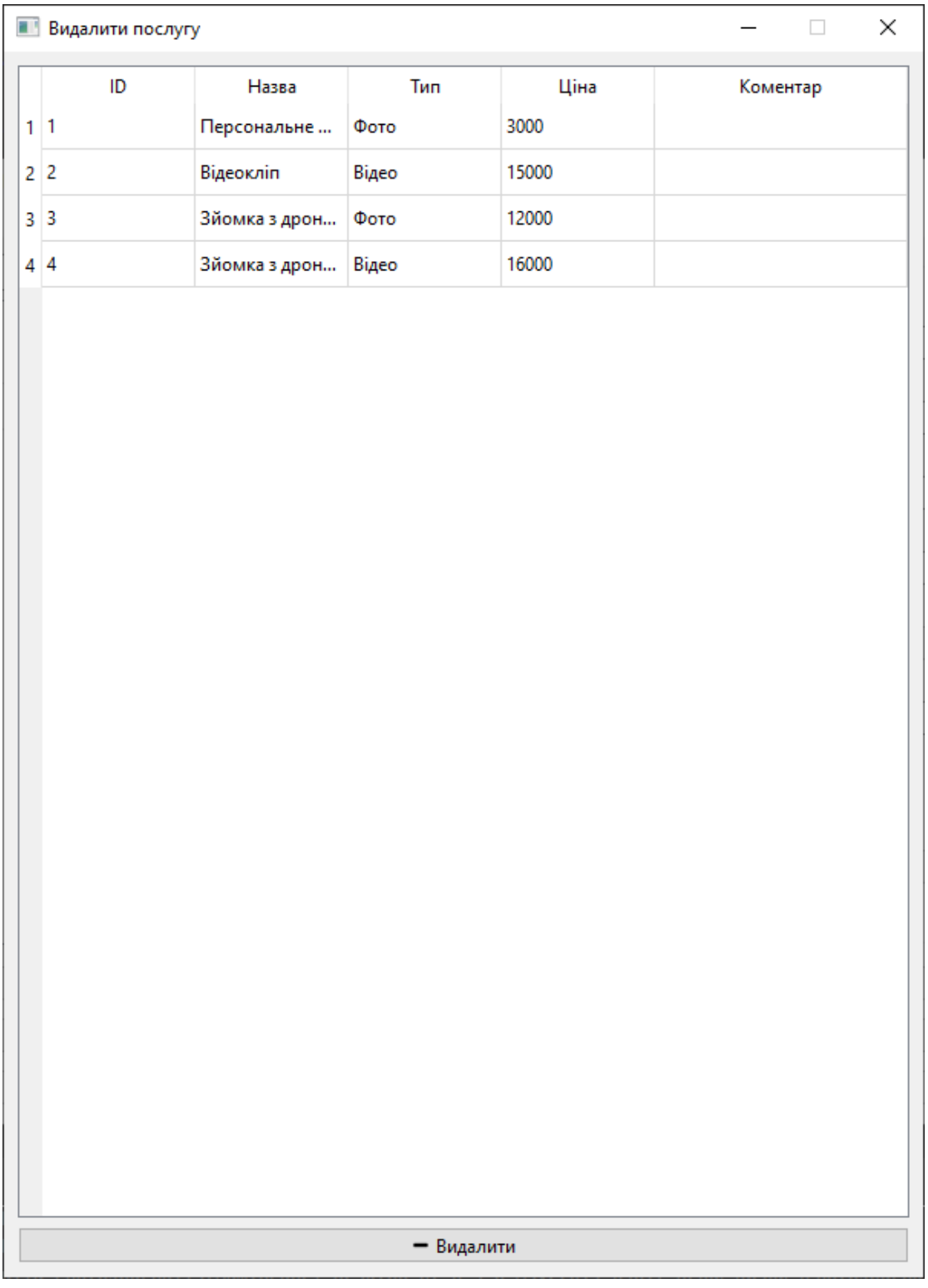

Рисунок 3.6 - Вікно для видалення або змінення обраної послуги

3.3 Контроль вхідних даних програми

У програмі для контролю вхідних даних було використано регулярні вираження класу QRegularExpression, приклад регулярного вираження та створення валідатору на введення даних лістинг 3.1.

Лістинг 3.1 – Приклад регулярного вираження для слів

```
QRegularExpression rx("[A-z А-я]+");
QValidator *text_validator = new QRegularExpressionValidator(rx,
this);
```

Лістинг 3.2 – Приклад регулярного вираження для номеру телефону

```
rx = QRegularExpression("[0-9]{12}");
QValidator *number_validator = new QRegularExpressionValidator(rx,
this);
```

Для виведення повідомлень про помилку або успіх було використано клас QMessageBox та види віконних повідомлень warning (для помилки) та information (для успіху). Приклад наведено у лістингу 3.2.

Лістинг 3.3 – Приклад виведення повідомлення помилки

```
QMessageBox::warning(this, "Ошибка", "Вы ввели не правильно номер телефона!");
```

Лістинг 3.3 – Приклад виведення повідомлення успіху

```
QMessageBox::information(this, "Успех", "Подключено!");
```

Лістинг 3.4 – Приклад виведення вікна з питанням про збереження

```
QMessageBox::question(this, "Сохранение", "Вы уверены, что хотите сохранить внесенные данные?", QMessageBox::Yes|QMessageBox::No)
```

При введенні некоректних значень програма повертає інформаційні повідомлення

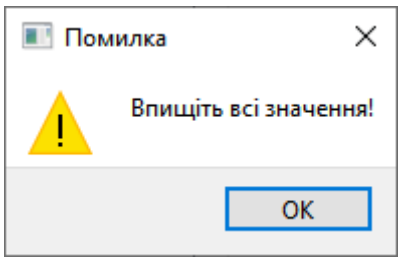

Рисунок 3.7 - Повідомлення про недостатню кількість введених значень

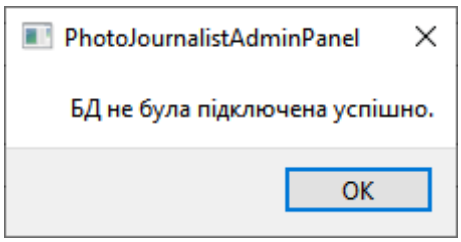

Рисунок 3.8 - Повідомлення про невдале підключення до бази даних

4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНИМ ІНТЕРФЕЙСОМ

Додавання нового замовлення

На початку програми користувача зустрічає головне віко. Для додавання нового замовлення необхідно натиснути на відповідну кнопку після чого відкриється вікно додавання нового замовлення.

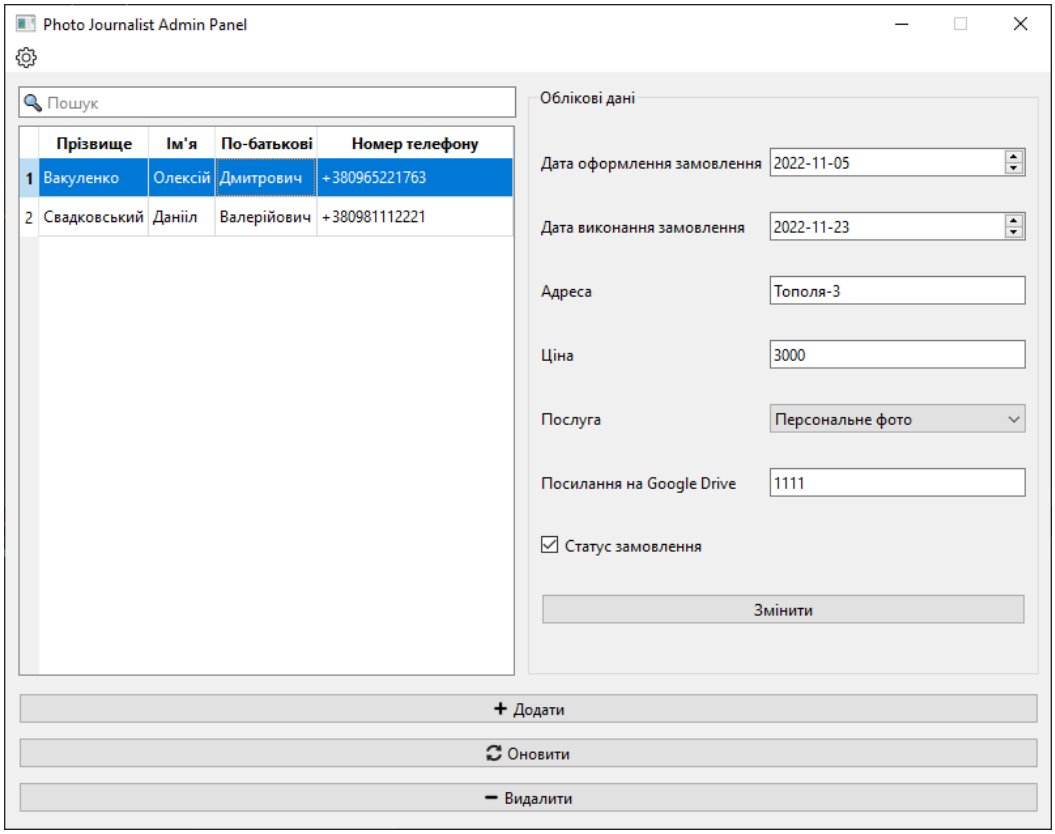

Рисунок 4.1 - Головне вікно програми

У відкритому вікні необхідно ввести замовника та обрати дату на яку виконати замовлення. Потім необхідно внести значення у поля адреса та обрати послугу з запропонованих у списку із послугами. Також можна додати додаткові значення до замовлення, такі як “Кінець виконання замовлення” та “Посилання на гугл диск” із фото,

для цього необхідно поставити галочку на флажку “Статсу замовлення”. Після введення всіх даних необхідно натиснути на кнопку “Додати” і у випадку якщо всі введені значення біли коректні то данні додадуться до бази даних, у зворотному випадку повернеться інформаційне вікно з помилкою. Після натискання на кнопку вікно закриється та знову відкриється вікно із замовленнями з оновленою таблицею.

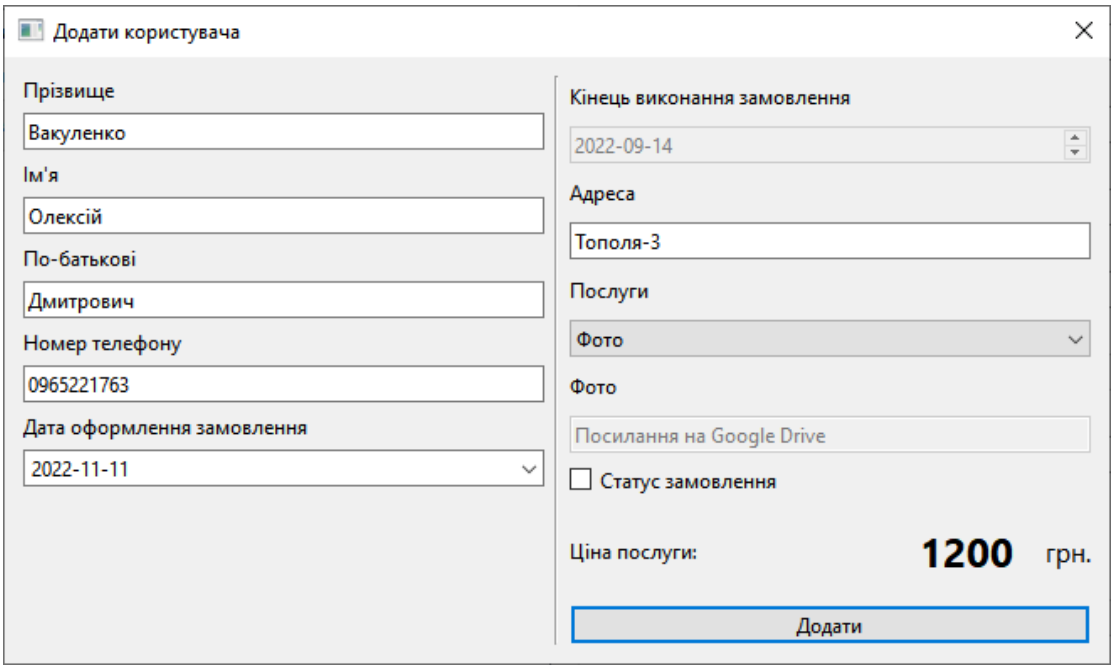

Рисунок 4.2 - Додавання нового замовлення без статусу замовлення

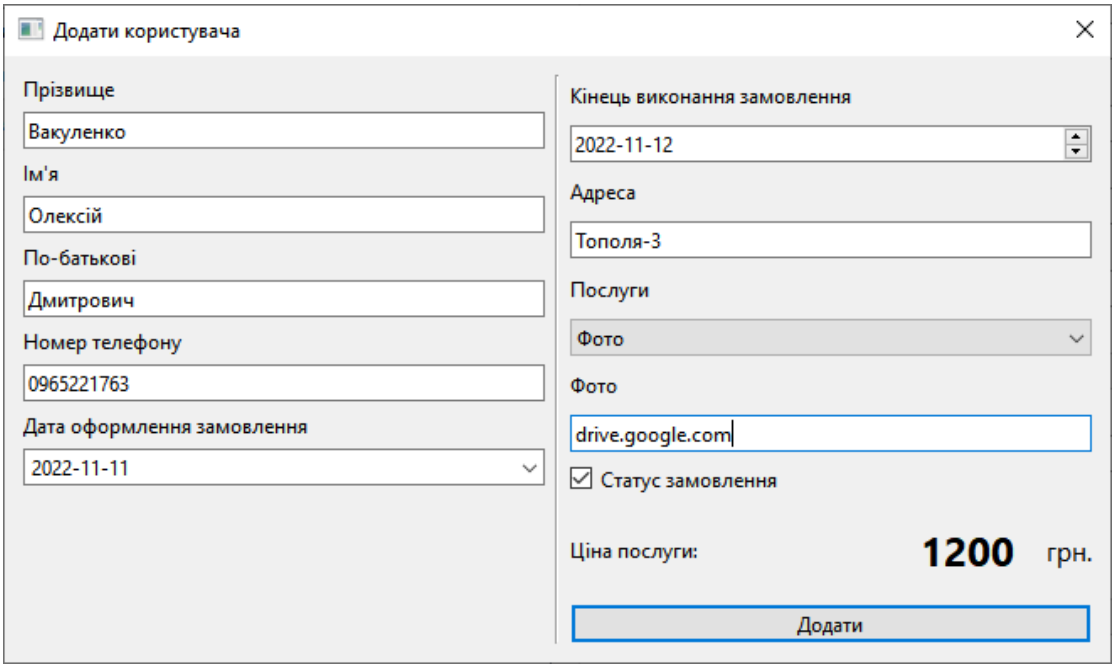

Рисунок 4.3 - додавання нового замовлення із статусом замовлення

Змінення даних про замовлення.

На початку, для змінення даних замовлення, необхідно обрати яке замовлення ми хочемо редагувати, після чого біль детально інформація про яке з’явиться у блоці “облікові дані”. Далі можна змінити будь-яке поля. Далі натискаємо кнопку “Змінити”, для того щаб зберегти зміни до бази даних. Після натиску на кнопку, у випадку якщо введені дані біли коректними, дані зміняться у баз даних та таблиця із даними про замовлення оновиться, у зворотному випадку програма поверне вікно із інформацією про помилку.

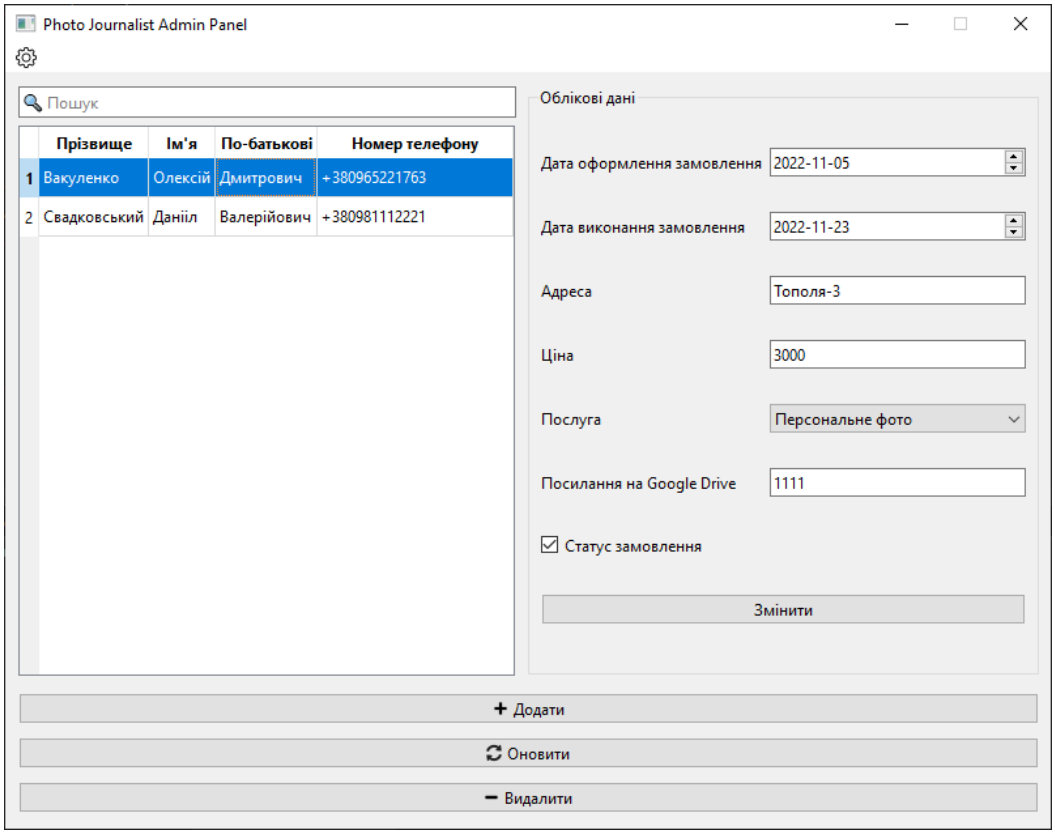

Рисунок 4.4 - Вікно із обраним елементом у таблиці із замовленнями

Редагування доступних послуг.

Для зміни послуг необхідно натиснути на іконку із шестернею після чого випаде список із ще двома кнопками. Перша кнопка служить для того, щоб додавати нову послугу. Друга, для того щоб видаляти або змінювати доступні послуги.

Натиснувши на першу кнопку відкриється вікно, у якому необхідно вписати данні нової послуги, після чого натиснути на кнопку “Додати”, після чого дані додадуться до бази якщо введені значення були корекними.

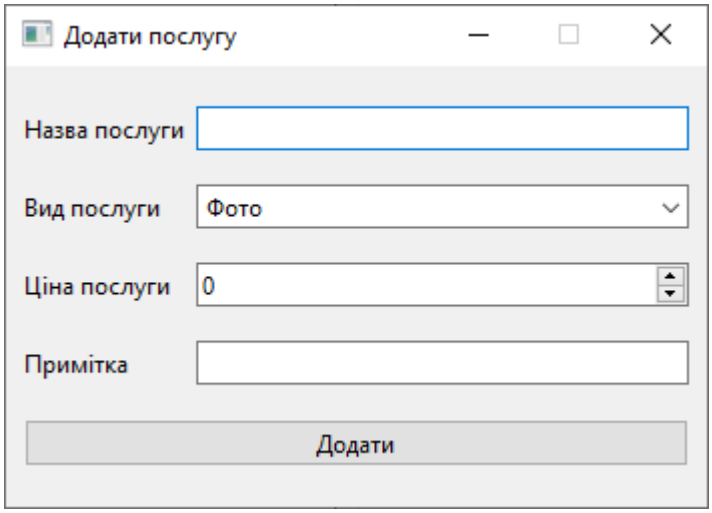

Рисунок 4.5 - Додавання послуги

Натиснувши на другу кнопку відкриється вікно редагування послуг.

Для редагування послуги необхідно змінити значення у блоку необхідної послуги натиснувши два рази на неї.

Для видалення послуги необхідно обрати елемент таблиці із послугами, після чого натиснути на кнопку видалення, після чого елемент видалиться із бали даних та таблиці.

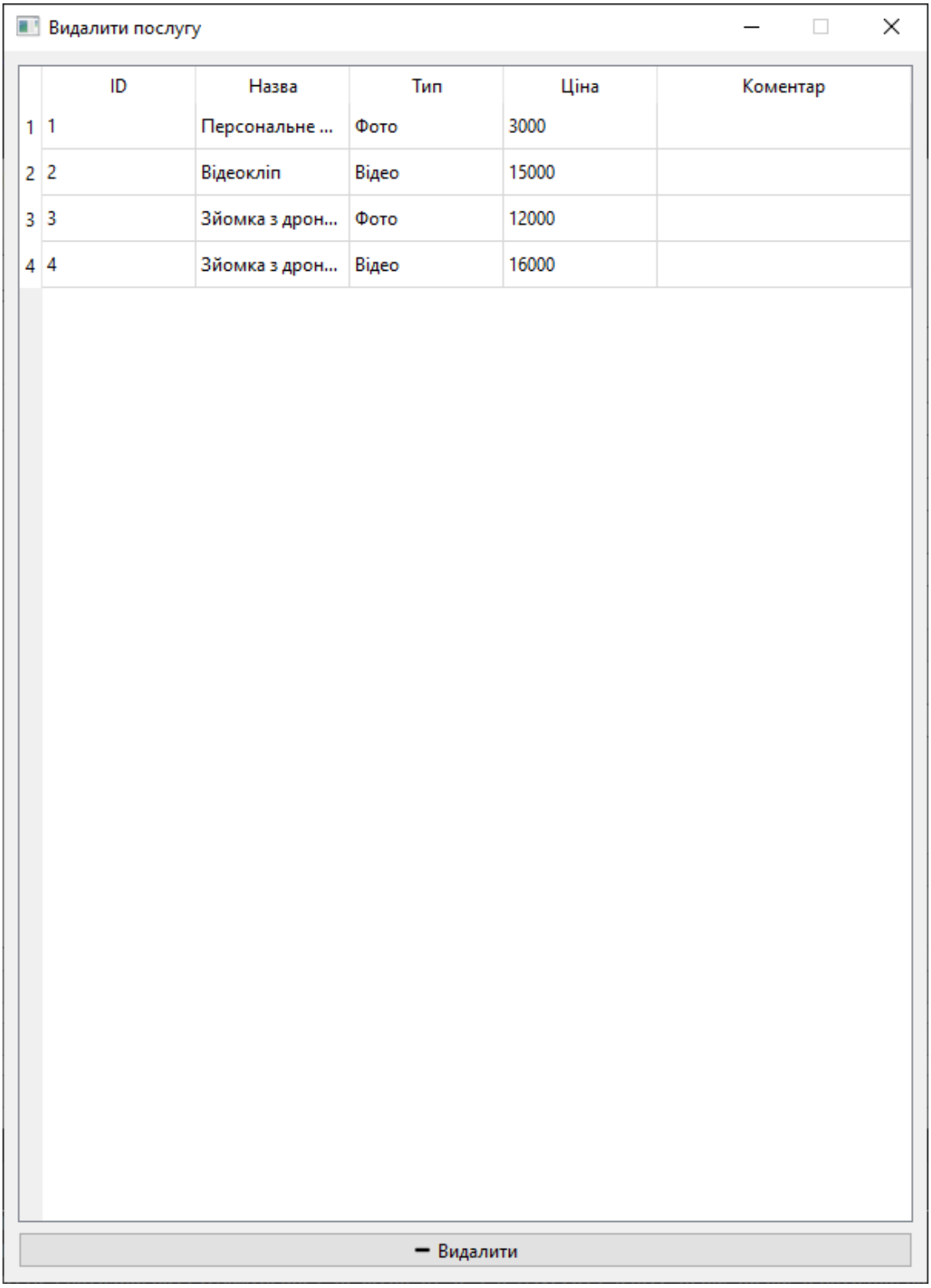

Рисунок 4.6 - Редагування та видалення послуг

Висновок

Під час виконання курсового проекту був розроблений програмний продукт який міг би знадобитися для людей які шукають якісь послуги фотографа. Завдяки цієї програми можна швидко знайти виконавця та оформити замовлення.

Також в програмі присутній легкий для розуміння інтерфейс та керування програмою. Для того щоб кожен міг знайти виконавця не зважаючи на вік та вміння володіти комп'ютером.

Також необхідно було зробити редагування послуг та замовлень для адміністраторів продукту.

Список використаних джерел

1 Офіційний сайт MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/dist/latest-v19.x/docs/api/.

2 Офіційний сайт розробника QT [Електронний ресурс] – Режим доступу до ресурсу: https://www.qt.io/product/ui–design–tools

3 Вікіпедія про використану СУБД [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/MySQL

4 Посібник: C/C++ Герберт Шилдт [Книга] – Режим доступу до ресурсу:https://www.bsuir.by/m/12_100229_1_98220.pdf

Додаток А

Код програми

Лістинг А.1 - Main.cpp

```
#include "DatabaseManager/databasemanager.h"
#include "AdminPanel/adminpanel.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    AdminPanel w;

    DatabaseManager::GetInstance()->ConnectToDatabase();

    w.show();

    return a.exec();
}
```

Лістинг А.2 - Deleteservices.cpp

```
#include "deleteservices.h"
#include "ui_deleteservices.h"

DeleteServices::DeleteServices(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::DeleteServices)
{
    ui->setupUi(this);

    DatabaseManager::GetInstance()->ConnectToDatabase();
    model = new QSqlTableModel();
    model->setTable("services");
    model->select();

    model->setHeaderData(0, Qt::Horizontal, tr("ID"));
    model->setHeaderData(1, Qt::Horizontal, tr("Назва"));
    model->setHeaderData(2, Qt::Horizontal, tr("Тип"));
    model->setHeaderData(3, Qt::Horizontal, tr("Ціна"));
    model->setHeaderData(4, Qt::Horizontal, tr("Коментар"));
```

```
    ui->ServicesTableView->setModel(model);
    ui->DeleteButton->setIcon(QIcon(":/icons/img/b_minus_icon.png"));
}

DeleteServices::~DeleteServices()
{
    delete ui;
}

void DeleteServices::on_DeleteButton_clicked()
{
    if(QMessageBox::question(this, "Видалення об'єкту", "Ви впевнені, що хочете видалити обрану послугу?",
                                                    QMessageBox::Yes|QMessageBox::No) ==
QMessageBox::Yes)
    {
        QModelIndex index =
ui->ServicesTableView->selectionModel()->currentIndex();
        QVariant value = index.sibling(index.row(), 0).data();
        QSqlQuery query("Delete from services where service_id = " +
value.toString());
        query.exec();

        model->select();
    }
}
```

Лістинг А.3 - Deleteservices.h

```
#ifndef DELETESERVICES_H
#define DELETESERVICES_H

#include <QMainWindow>
#include <DatabaseManager/databasemanager.h>

namespace Ui {
class DeleteServices;
}

class DeleteServices : public QMainWindow
{
```

```
    Q_OBJECT

public:
    explicit DeleteServices(QWidget *parent = nullptr);
    ~DeleteServices();

private slots:
    void on_DeleteButton_clicked();

private:
    Ui::DeleteServices *ui;
    QSqlTableModel *model;


};

#endif // DELETESERVICES_H
```

Лістинг А.4 - Databseoptions.cpp

```
#include "databaseoptions.h"
#include "ui_databaseoptions.h"

DataBaseOptions::DataBaseOptions(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::DataBaseOptions)
{
    ui->setupUi(this);
}

DataBaseOptions::~DataBaseOptions()
{
    delete ui;
}

void DataBaseOptions::on_ApplyButton_clicked()
{

DatabaseManager::GetInstance()->editDatabase(ui->HostnameLineEdit->text(),

ui->UserNameLineEdit->text(),

ui->PasswordLineEdit->text(),
```

```
                                                    ui->DBNameLineEdit->text());
    if(DatabaseManager::GetInstance()->checkDatabaseConnection())
    DataBaseOptions::~DataBaseOptions();
}
```

Лістинг А.5 - Databaseoptions.h

```
#ifndef DATABASEOPTIONS_H
#define DATABASEOPTIONS_H

#include <QMainWindow>
#include "DatabaseManager/databasemanager.h"

namespace Ui {
class DataBaseOptions;
}

class DataBaseOptions : public QMainWindow
{
    Q_OBJECT

public:
    explicit DataBaseOptions(QWidget *parent = nullptr);
    ~DataBaseOptions();

private slots:
    void on_ApplyButton_clicked();

private:
    Ui::DataBaseOptions *ui;
};

#endif // DATABASEOPTIONS_H
```

Лістинг А.6 - Databasemanager.cpp

```
#include "databasemanager.h"

DatabaseManager *DatabaseManager::GetInstance()
{
    if(s_Instance==nullptr){
```

```
        s_Instance = new DatabaseManager();
    }
    return s_Instance;
}

DatabaseManager* DatabaseManager::s_Instance;

DatabaseManager::DatabaseManager()
{
}

void DatabaseManager::ConnectToDatabase()
{
    if(!m_db.open())
    {
        m_db = QSqlDatabase::addDatabase("QODBC");
        m_db.setHostName("192.168.0.77");
        m_db.setPort(3306);
        m_db.setUserName("root");
        m_db.setPassword("123123");
        m_db.setDatabaseName("coursework");
        checkDatabaseConnection();
    }
    return;
}

bool DatabaseManager::checkDatabaseConnection()
{
    QMessageBox *message = new QMessageBox();
    if(m_db.open())
    {
        message->setText("БД підключена успішно.");
        message->exec();
        return true;
    }
    else
    {
        message->setText("БД не була підключена успішн
o.");
        message->exec();
        return false;
    }
}
```

```
void DatabaseManager::editDatabase(QString hostname, QString username,
QString password, QString databaseName)
{
    m_db.setHostName(hostname);
    m_db.setUserName(username);
    m_db.setPassword(password);
    m_db.setDatabaseName(databaseName);
}
```

Лістинг А.7 - Databsemanager.h

```
#ifndef DATABASEMANAGER_H
#define DATABASEMANAGER_H
#include <QtSql>
#include <QSqlDatabase>
#include <qmessagebox.h>

class DatabaseManager
{
private:
    QSqlDatabase m_db;
    DatabaseManager();
    static DatabaseManager* s_Instance;

public:
    DatabaseManager(DatabaseManager &other) = delete;
    void operator=(const DatabaseManager &) = delete;

    static DatabaseManager *GetInstance();

    void ConnectToDatabase();
    bool checkDatabaseConnection();
    void editDatabase(QString, QString, QString, QString);
};

#endif // DATABASEMANAGER_H
```

Лістинг А.8 - Adminpanel.cpp

```
#include "AdminPanel/adminpanel.h"

AdminPanel::AdminPanel(QWidget *parent) :
```

```
    QMainWindow(parent),
    ui(new Ui::AdminPanel)
{
    ui->setupUi(this);

    DatabaseManager::GetInstance()->ConnectToDatabase();
    model = new QSqlQueryModel();


    WidgetSettings();
    initResources();
    updateTable();
}

AdminPanel::~AdminPanel()
{
    delete ui;
}

void AdminPanel::initResources()
{
    ui->AddButton->setIcon(QIcon(":/icons/img/b_plus_icon.png"));
    ui->DeleteButton->setIcon(QIcon(":/icons/img/b_minus_icon.png"));
    ui->UpdateButton->setIcon(QIcon(":/icons/img/b_refresh_icon.png"));

    ui->OptionsMenu->setIcon(QIcon(":/icons/img/b_gear_icon.png"));
    ui->ProfileAction->setIcon(QIcon(":/icons/img/b_user_icon.png"));
    ui->OptionsAction->setIcon(QIcon(":/icons/img/b_gear_icon.png"));
}

void AdminPanel::updateTable(QString arg)
{
    if(arg != "")
    {
        QString request = "SELECT * FROM users WHERE users_surname LIKE
'%%1%' or users_name LIKE '%%1%' or users_middle_name LIKE '%%1%' or
phone_number LIKE '%%1%'";
        request = request.arg(arg);
        model->setQuery(request);
    }
    else
    {
        model->setQuery("SELECT * FROM users");
    }
```

```
    model->setHeaderData(0, Qt::Horizontal, tr("ID"));
    model->setHeaderData(1, Qt::Horizontal, tr("Прізвище"));
    model->setHeaderData(2, Qt::Horizontal, tr("Ім'я"));
    model->setHeaderData(3, Qt::Horizontal, tr("По-батькові"));
    model->setHeaderData(4, Qt::Horizontal, tr("Номер телефону
"));

    ui->UsersTable->setModel(model);
    ui->UsersTable->hideColumn(0);
    initCombobox();
}

void AdminPanel::WidgetSettings()
{

ui->UsersTable->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeT
oContents);
    ui->UsersTable->horizontalHeader()->setStretchLastSection(true);
}

void AdminPanel::on_AddButton_clicked()
{
    AddUserData window;
    window.setModal(true);
    window.exec();

    updateTable();
}

void AdminPanel::on_AddService_triggered()
{
    AddService *window = new AddService();
    window->show();
}

void AdminPanel::on_DBOptionsAction_triggered()
{
    DataBaseOptions *window = new DataBaseOptions();
    window->show();
}

void AdminPanel::on_DeleteButton_clicked()
{
```

```
    if(QMessageBox::question(this, "Видалення об'єкту", "Ви впевнені, що хочете видалити обраного користувача?",
                                                    QMessageBox::Yes|QMessageBox::No) ==
QMessageBox::Yes)
    {
        QModelIndex index = ui->UsersTable->selectionModel()->currentIndex();
        QString id = index.sibling(index.row(), 0).data().toString();
        QSqlQuery query;

        query.exec("Delete from order_data where users_ID = " + id);
        query.exec("Delete from users where users_ID = " + id);

        updateTable();
    }
}

void AdminPanel::on_UpdateButton_clicked()
{
    updateTable();
}

void AdminPanel::on_DeleteService_triggered()
{
    DeleteServices *window = new DeleteServices();
    window->show();
}

void AdminPanel::initCombobox()
{
    QStringList status;

    QSqlQuery query;
    query.exec("select * from services");
    while (query.next())
    {
        QSqlRecord recorder = query.record();
        status.append(recorder.value("name_service").toString());
    }

    ui->ServiceComboBox->clear();
    ui->ServiceComboBox->addItems(status);
}
```

```
void AdminPanel::on_UsersTable_clicked(const QModelIndex &index)
{
    QSqlQuery query;
    query.prepare("select * from order_data INNER JOIN services ON
order_data.service_ID = services.service_ID where users_ID = " +
ui->UsersTable->model()->index(index.row(),0).data().toString());
    query.exec();
    query.next();

    QSqlRecord record = query.record();


ui->StartDate->setDate(QDate::fromString(record.value("creation_date").toStri
ng(), "yyyy-MM-dd"));

ui->ServiceComboBox->setCurrentText(record.value("name_service").toString());
    ui->AdressLineEdit->setText(record.value("adress").toString());
    ui->PriceLineEdit->setText(record.value("order_price").toString());
    ui->statusCheckBox->setChecked(record.value("order_status").toBool());

    if(ui->statusCheckBox->isChecked())
    {
        ui->EndDate->setEnabled(true);
        ui->PhotoPathLineEdit->setEnabled(true);

ui->EndDate->setDate(QDate::fromString(record.value("finalization_date").toSt
ring(), "yyyy-MM-dd"));
        ui->PhotoPathLineEdit->setText(record.value("img").toString());
    }
    else
    {
        ui->EndDate->setEnabled(false);
        ui->PhotoPathLineEdit->setEnabled(false);
    }
}

bool AdminPanel::linesIsNotEmpty()
{
    return !ui->StartDate->text().isEmpty() &&
           !ui->EndDate->text().isEmpty() &&
           !ui->PriceLineEdit->text().isEmpty() &&
           !ui->PhotoPathLineEdit->text().isEmpty() &&
           !ui->AdressLineEdit->text().isEmpty() &&
           !ui->ServiceComboBox->itemText(-1).contains("П о с л у г и") &&
```

```
            !ui->EndDate->date().isNull() &&
            !ui->PhotoPathLineEdit->text().isEmpty();
}

void AdminPanel::on_EditButton_clicked()
{
    if(linesIsNotEmpty())
    {
        int serviceId = 0;
        int userId = 0;

        QSqlQuery query;
        query.exec("SELECT service_ID from services where type_service = " +
ui->ServiceComboBox->currentText());
        while(query.next())
        {
            QSqlRecord recorder = query.record();
            serviceId = recorder.value("service_ID").toInt();
        }

        QModelIndex index = ui->UsersTable->selectionModel()->currentIndex();
        userId = index.sibling(index.row(), 0).data().toInt();


        QString request = "UPDATE order_data SET creation_date = '%1',
finalization_date = '%2', img = '%3', adress = '%4', order_price = '%5',
service_ID = %6 where users_ID = %7";
        request = request.arg(ui->StartDate->text(), ui->EndDate->text(),
ui->PhotoPathLineEdit->text(), ui->AdressLineEdit->text(),
ui->PriceLineEdit->text(), QString::number(serviceId),
QString::number(userId));

        if(query.exec(request))
            QMessageBox::warning(this, "Вдало", "Данні були змі
нені!");
        else
            QMessageBox::warning(this, "Помилка", "Помилка за
просу!");
    }
    else
    {
        QMessageBox::warning(this, "Помилка", "Впищіть всі зн
ачення!");
    }
```

```
}


void AdminPanel::on_SearchLineEdit_textChanged(const QString &arg1)
{
    updateTable(arg1);
}
```

Лістинг А.9 - Adminpanel.h

```
#ifndef ADMINPANEL_H
#define ADMINPANEL_H

#include <QMainWindow>

#include "DatabaseManager/databasemanager.h"
#include "AddUserData/adduserdata.h"
#include "AddService/addservice.h"
#include "DBOptions/databaseoptions.h"
#include "DatabaseManager/databasemanager.h"
#include <DeleteServices/deleteservices.h>

#include "qsqltablemodel.h"
#include "ui_adminpanel.h"
#include "qsqlquery.h"
#include "qdebug.h"

namespace Ui {
class AdminPanel;
}

class AdminPanel : public QMainWindow
{
    Q_OBJECT

public:
    explicit AdminPanel(QWidget *parent = nullptr);
    ~AdminPanel();

private slots:
    void on_AddButton_clicked();
    void on_AddService_triggered();
    void on_DBOptionsAction_triggered();
    void on_DeleteButton_clicked();
```

```
    void on_UpdateButton_clicked();


    void on_DeleteService_triggered();
    void on_UsersTable_clicked(const QModelIndex &index);

    void on_EditButton_clicked();

    void on_SearchLineEdit_textChanged(const QString &arg1);

private:
    Ui::AdminPanel *ui;
    QSqlQueryModel *model;

    void initResources();
    void updateTable(QString arg = "");
    void initCombobox();
    void WidgetSettings();
    bool linesIsNotEmpty();
};

#endif // ADMINPANEL_H
```

Лістинг А.10 - Adduserdata.cpp

```
#include "adduserdata.h"
#include "ui_adduserdata.h"

AddUserData::AddUserData(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AddUserData)
{
    ui->setupUi(this);
    ui->EndOrderDate->setEnabled(false);
    ui->PhotoPathLineEdit->setEnabled(false);

    initComboBox();
    setValidFields();
}

AddUserData::~AddUserData()
{
    delete ui;
```

```
}

void AddUserData::on_StatusCheckBox_toggled(bool checked)
{
    ui->EndOrderDate->setEnabled(checked);
    ui->PhotoPathLineEdit->setEnabled(checked);
}

void AddUserData::setValidFields()
{
    QRegularExpression rx("[A-z А-я Ё ё ЄєЇїІі]+");
    QValidator *validator = new QRegularExpressionValidator(rx, this);

    ui->NameLineEdit->setValidator(validator);
    ui->MiddleNameLineEdit->setValidator(validator);
    ui->SurnameLineEdit->setValidator(validator);

    rx.setPattern("^[0-9]{10}$");
    QValidator *phoneValidator = new QRegularExpressionValidator(rx, this);
    ui->PhoneNumberLineEdit->setValidator(phoneValidator);
}

void AddUserData::initComboBox()
{
    QStringList status;

    QSqlQuery query;
    query.exec("select * from services");
    while (query.next())
    {
        QSqlRecord recorder = query.record();
        status.append(recorder.value("name_service").toString());
    }

    ui->ServiceComboBox->addItems(status);
}

bool AddUserData::checkIncorrectFields()
{
    bool correct = !ui->NameLineEdit->text().isEmpty() &&
                !ui->MiddleNameLineEdit->text().isEmpty() &&
                !ui->SurnameLineEdit->text().isEmpty() &&
                !ui->PhoneNumberLineEdit->text().isEmpty() &&
                !ui->AdressLineEdit->text().isEmpty() &&
```

```
                !(ui->ServiceComboBox->currentText() == "П о с л у г и") &&
                !ui->PriceLabel_2->text().isEmpty() &&
                !ui->StartDateEdit->date().isNull();

    if(ui->StatusCheckBox->isChecked())
    {
        correct = correct &&
                  !ui->EndOrderDate->date().isNull() &&
                  (ui->EndOrderDate->date() >= ui->StartDateEdit->date()) &&
                  !ui->PhotoPathLineEdit->text().isEmpty();
    }

    return correct;
}

void AddUserData::addUser()
{
    QSqlQuery query_user;
    query_user.prepare("INSERT INTO users(users_name, users_middle_name,
users_surname, phone_number)"
                       "VALUES(:name, :midname, :surname, :p_num)");
    query_user.bindValue(":name",          ui->NameLineEdit->text());
    query_user.bindValue(":midname",       ui->MiddleNameLineEdit->text());
    query_user.bindValue(":surname",       ui->SurnameLineEdit->text());
    query_user.bindValue(":p_num", "+38" + ui->PhoneNumberLineEdit->text());
    query_user.exec();

    int user_id = query_user.lastInsertId().toUInt();

    QSqlQuery order_query;
    order_query.prepare("INSERT INTO order_data(creation_date,
finalization_date, img, adress, order_price, order_status, users_ID,
service_ID) "

"VALUES(:s_date, :f_date, :img, :adress, :price, :status, :user_id, :service)
");
    order_query.bindValue(":s_date", ui->StartDateEdit->text());
    /*order_query.bindValue(":f_date", "");
    order_query.bindValue(":img", "");*/

    order_query.bindValue(":adress", ui->AdressLineEdit->text());
    order_query.bindValue(":price",  ui->PriceLabel_2->text().toInt());
    order_query.bindValue(":status", ui->StatusCheckBox->isChecked());
    order_query.bindValue(":user_id", user_id);
```

```
    order_query.bindValue(":service", service_id);


    if(ui->StatusCheckBox->isChecked())
    {
        order_query.bindValue(":f_date", ui->EndOrderDate->text());
        order_query.bindValue(":img", ui->PhotoPathLineEdit->text());
    }

    order_query.exec();
}


void AddUserData::on_AddUserButton_clicked()
{
    qDebug() << checkIncorrectFields();

    if(checkIncorrectFields())
    {
        addUser();
        this->close();
    }
}

void AddUserData::on_ServiceComboBox_textActivated(const QString &arg1)
{
    QSqlQuery query;
    query.prepare("SELECT service_ID, service_price FROM services WHERE
name_service = :name;");
    query.bindValue(":name", arg1);
    query.exec();
    query.next();

    QSqlRecord record = query.record();

    ui->PriceLabel_2->setText(record.value("service_price").toString());
    service_id = record.value("service_ID").toInt();
}
```

Лістинг А.11 - Adduserdata.h

```
#ifndef ADDUSERDATA_H
#define ADDUSERDATA_H
```

```
#include <QMainWindow>
#include "qsqlquerymodel.h"
#include "qsqlquery.h"
#include "AddService/addservice.h"
#include <QCloseEvent>

namespace Ui {
class AddUserData;
}

class AddUserData : public QDialog
{
    Q_OBJECT

public:
    explicit AddUserData(QWidget *parent = nullptr);
    ~AddUserData();

private slots:
    void on_StatusCheckBox_toggled(bool checked);
    void on_AddUserButton_clicked();

    void on_ServiceComboBox_textActivated(const QString &arg1);

private:
    Ui::AddUserData *ui;
    void setValidFields();
    void initComboBox();
    bool checkIncorrectFields();
    void addUser();
    QString calculatePrice(int);

    int service_id = 0;
};

#endif // ADDUSERDATA_H
```

Лістинг А.12 - Addservice.cpp

```
#include "addservice.h"
#include "ui_addservice.h"

AddService::AddService(QWidget *parent) :
```

```
    QMainWindow(parent),
    ui(new Ui::AddService)
{
    ui->setupUi(this);

    initCombobox();
}

AddService::~AddService()
{
    delete ui;
}

void AddService::initCombobox()
{
    QSqlQuery query;
    query.exec("SELECT DISTINCT type_service FROM services");

    QStringList items;
    while(query.next())
    {
        QSqlRecord recorder = query.record();
        items.append(recorder.value("type_service").toString());
    }

    ui->ServiceType->addItems(items);
}

void AddService::on_AddServiceButton_clicked()
{
    if(!CheckFields())
        return;

    qDebug() << ui->ServiceType->currentText();

    QSqlQuery query;
    QString request("INSERT INTO services(name_service, type_service,
service_price, commentary) VALUES('%1', '%2', %3, '%4')");
    request = request.arg(ui->ServiceTitleLineEdit->text(),
ui->ServiceType->currentText(), ui->PriceSpinBox->text(),
ui->CommentaryTextEdit->text());

    if(query.exec(request))
```

```
        QMessageBox::information(this, "Успішно", "Додавання
послуги пройшло успішно", QMessageBox::StandardButton());

    this->close();
}

bool AddService::CheckFields()
{
    return !ui->ServiceTitleLineEdit->text().isEmpty() &&
           !ui->PriceSpinBox->text().isEmpty();
}
```

Лістинг А.13 - Addservice.h

```
#ifndef ADDSERVICE_H
#define ADDSERVICE_H

#include "qsqlquery.h"
#include "DatabaseManager/databasemanager.h"
#include <QMainWindow>

namespace Ui {
class AddService;
}

class AddService : public QMainWindow
{
    Q_OBJECT

public:
    explicit AddService(QWidget *parent = nullptr);
    ~AddService();

private slots:
    void on_AddServiceButton_clicked();

private:
    Ui::AddService *ui;
    void initCombobox();
    bool CheckFields();
};
```

```
#endif // ADDSERVICE_H
```

Додаток Б

Скрипт створення бази даних

Лістинг Б.1 - створення таблиці user

```
CREATE TABLE if not exists users(
    users_ID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT ,
    users_surname varchar(32) NOT NULL,
    users_name varchar(32) NOT NULL,
    users_middle_name varchar(32) NOT NULL,
    phone_number varchar(13) NOT NULL
);
```

Лістинг Б.2 - створення таблиці services

```
CREATE TABLE if not exists services(
    service_ID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name_service varchar(32) NOT NULL,
    type_service varchar(32) NOT NULL,
    service_price decimal NOT NULL,
    commentary text
);
```

Лістинг Б.3 - створення таблиці order_data

```
CREATE TABLE if not exists order_data(
    order_ID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    creation_date date NOT NULL,
    finalization_date date DEFAULT NULL,
    img VARCHAR(128) DEFAULT NULL,
    adress varchar(128) NOT NULL,
    order_price INT NOT NULL,
    order_status BOOLEAN NOT NULL,
    users_ID INT NOT NULL,
    service_ID INT NOT NULL,
    FOREIGN KEY(users_ID) REFERENCES users(users_ID) ON DELETE CASCADE,
    FOREIGN KEY(service_ID) REFERENCES services(service_ID) ON DELETE CASCADE
);
```

Додаток В

Вхідні дані

	* users_ID int	* users_surname varchar(32)	* users_name varchar(32)	* users_middle_name varchar(32)	* phone_number varchar(13)
1	13	Вакуленко	Олексій	Дмитрович	+380965221763
2	14	Свадковський	Даніїл	Валерійович	+380961234567

Рисунок В.1 - Користувачі

	* service_ID int	* name_service varchar(32)	* type_service varchar(32)	* service_price decimal	commentary text(65535)
1	1	Фото	Фото	1200	
2	2	Відео	Відео	1500	
3	3	Шлюбне фото	Фото	12000	
4	4	Зйомка з дрону	Відео	1500	

Рисунок В.2 - Послуги

	* order_ID int	* creation_date date	finalization_date date	img varchar(128)	* adress varchar(128)	* order_price int	* order_status tinyint	* users_ID int	* service_ID int
1	5	2022-11-11	(NULL)	(NULL)	Тополя-3	15001	0	13	4
2	6	2022-11-12	2022-11-24	url.com	Тополя-1	12000	1	14	3

Рисунок В.3 - Замовлення