

Міністерство освіти і науки України
ДВНЗ «Донецький національний технічний університет»

Кафедра комп'ютерної інженерії

Пояснювальна записка
до курсової роботи
з дисципліни «Об'єктно-орієнтоване програмування»
на тему
«Об'єктно-орієнтоване моделювання програми керування
клієнтами інтернет провайдера»

Виконав студент

групи КІ-16

напряму підготовки «Комп'ютерна інженерія»

Котлов К.О.

Керівники

к.т.н., доц. кафедри КІ *Цололо С.О.*

ст. викл. кафедри КІ *Дікова Ю.Л.*

Оцінка

Національна шкала _____

Кількість балів: _____

Члени комісії

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

Покровськ, 2017

ЗМІСТ

Вступ.....	3
1. Аналіз завдання	4
2. Розробка бібліотеки класів.....	6
2.1. Проектування системи класів	6
2.2. Формалізація опису класів у вигляді діаграми класів.....	12
2.3. Програмна реалізація класів.	13
3. Розробка інтерфейсу програми.....	15
3.1.Опис користувацького інтерфейсу.....	15
3.2. Програмна реалізація інтерфейсу.....	25
Висновки.....	26
Список літератури	27
Додаток А – Діаграма класів.....	28
Додаток Б – Лістинг реалізації класів.....	31
Додаток В – Лістинг реалізації інтерфейсу.....	46

ВСТУП

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм[1].

Метою курсової роботи є опрацювання об'єктно-орієнтованої парадигми програмування. В якості завдання була вибрана програма керування клієнтами інтернет-провайдера, яка повинна включати до себе усі концепції ООП, такі як:

- інкапсуляція,
- наслідування,
- поліморфізм,
- композиція (наповнення),
- перевантаження,
- перевизначення.

Основний модуль програми був написаний на мові програмування C++, в якості графічного інтерфейсу користувача був вибраний крос-платформний фреймворк Qt.

1. АНАЛІЗ ЗАВДАННЯ

У курсовій роботі мені належало виконати програму управління клієнтами інтернет провайдера. Обов'язковою вимогою було використання усіх концепцій об'єктно-орієнтованого програмування, а саме:

- інкапсуляція,
- наслідування,
- поліморфізм,
- композиція (наповнення),
- перевантаження,
- перевизначення.

Крім того, були потрібні перевірки коректності даних, що вводилися. Ця вимога була реалізована за допомогою механізму регулярних виразів. У кожне поле можна ввести тільки кирилицю, перша буква обов'язково заголовна, а обійти таку перевірку неможливо. У програмі реалізований наступний функціонал:

- додавання нового облікового запису,
- редагування існуючого облікового запису,
- видалення облікового запису,
- фінансові операції (підключення, відключення(наприклад за власним бажанням), відключення за борги, редагування балансу з датою останнього платежу),
- виведення даних з таблиці в текстові файли. Вивести можна історію платежів, список боржників, список клієнтів з сортуванням по місту, прайс-лист на послуги.

Програма була написана з використанням крос-платформного фреймворка Qt [1]. Крім того, програма використовує віджети - графічний інтерфейс користувача. Одного разу написана програма з використанням цього фреймворка здатна працювати в наступних операційних системах:

- Windows,
- UNIX-like (Linux, *BSD),
- iOS,
- Android,
- Windows Phone.

В якості інструменту зберігання даних була вибрана реляційна база даних SQLite [2]. Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

2. РОЗРОБКА БІБЛІОТЕКИ КЛАСІВ

2.1 Проектування системи класів

Програма повинна уміти прочитати, додати, записати, відредагувати і видалити запис з бази даних(далі - БД). Для цього були написані наступні класи:

- AddCustomer - клас, що додає новий запис у БД,
- AddMoney - клас для роботи з фінансовими операціями,
- Database - ключовий клас для роботи з базою даних,
- EditCustomer - клас, що дозволяє редагувати записи,
- MainWindow - клас, що займається відображенням таблиці облікових записів і відповідних кнопок(додати, видалити і так далі),
- OutputDocuments - клас для виведення даних з таблиці.

Клас Database

Цей клас включає методи відкриття, додавання і витягання даних з БД. Для початку роботи треба підключитися до бази даних, для цього є 2 методи: перший шукає базу даних в директорії з виконуваним файлом. На випадок якщо БД там не виявилось, є перевантажений метод відкриття БД, який приймає як аргумент рядок, в якому повинен вказуватися шлях до БД. Метод додавання у базу даних приймає як аргумент контейнер QVariantList. У самому методі вміст контейнера розбивається на рядки і числа і додається у базу даних за допомогою оператора SQL INSERT. Метод очищення БД виконується за допомогою оператора SQL DELETE. Методи витягання даних з БД повертають контейнер QList, в якому зберігаються дані типу QString, передбачено 3 методи:

- метод отримання даних про історію платежів,
- метод отримання даних про боржників,
- метод отримання списку даних клієнтів з потрібного міста, метод приймає в якості аргументу назву міста.

Структура бази даних

База даних являє собою таблицю Customers, структуру якої розглянемо в таблиці 1.

Таблиця 1 – структура бази даних

Назва стовпця	Тип даних	Призначення
ID	INTEGER	Первинний ключ
SURNAME	TEXT	Прізвище
NAME	TEXT	Ім'я
PATRONYMIC	TEXT	По батькові
BALANCE	INTEGER	Баланс
CITY	TEXT	Місто
STREET	TEXT	Вулиця
BUILDING	INTEGER	Будинок
APARTMENT	INTEGER	Квартира
TYPE	TEXT	Тип підключення
TARIFF	TEXT	Тарифний план
CONNECTED	TEXT	Дата підключення
LAST_PAY	TEXT	Останній платіж
STATUS	TEXT	Статус

Для кожного поля встановлено властивість NOT NULL, це потрібно для того, щоб в БД не можна було записати пусте значення.

Для роботи з класом Database потрібні були такі класи:

- QSqlDatabase - для роботи з БД,
- QSqlQuery - для виконання запитів SQL,
- QDebug - для виведення налагоджувальних повідомлень в консоль налагодження,
- QSqlError - для виведення помилок, пов'язаних з SQL,
- QSqlRecord - для читання записів з БД,
- QApplication - для отримання директорії, в якому знаходиться виконуваний файл.

Клас успадкований від QObject.

Клас AddCustomer

Цей клас включає методи перевірки коректності даних і додавання нового запису в базу даних. Після введення всіх даних виконується перевірка, і якщо щось було введено не так, програма попросить перевірити введені дані. Дані з форми приймаються за допомогою класів:

- QLineEdit - для імені, прізвища, по батькові, міста і вулиці.
- QSpinBox - для номера будинку та номера квартири
- QComboBox - для тарифу, статусу і типу підключення.

Програма використовує механізм обробки подій - сигнали і слоти [3]. Сигнали і слоти потрібні для комунікації між об'єктами. Для того, щоб використовувати сигнали і слоти, потрібно успадкувати клас від класу QObject і позначити успадкований клас макросом Q_OBJECT. Для з'єднання об'єктів використовується метод

QObject :: connect (відправник, сигнал (), одержувач, слот ()).

Сигнал є звичайною функцією, але в класі він розташований в секції slots, сигнали в секції signals.

Розглянемо застосування сигналів і слотів в даному класі:

Є метод додавання запису в базу даних, але перед тим, як додати запис, потрібно переконатися, що перевірки коректності проведені. Додавання запису здійснюється натисканням на кнопку. У кнопки (QPushButton) є свої сигнали, їх багато, але нас цікавить сигнал clicked(bool) - натискання вироблено. Методом connect ми з'єднуємо сигнал кнопки clicked (bool) зі слотом check (), який перевірить, що всі дані введені правильно і в свою чергу викличе метод додавання запису в БД. Виглядає це так:

```
connect(ui->pbApply, SIGNAL(clicked(bool), this, SLOT(check()));
```

Після проходження всіх перевірок, дані формуються в контейнер QVariantList, і викликається метод класу Database - insertIntoTable, який прийме як аргумент тільки що сформований контейнер. Також випускається сигнал void applyChanges(), який потрібен для того, щоб повідомити

табличне представлення в класі MainWindow про те, що було додано новий запис і потрібно оновити дані в таблиці.

Для роботи з класом AddCustomer потрібні були такі класи:

- QDateTime - для отримання часу додавання запису
- QMessageBox – попередження про некоректне введення
- Database - для додавання нового запису в БД.

Клас успадкований від QDialog.

Клас EditCustomer

Цей клас включає методи перевірки коректності даних і редагування обраного запису. Для редагування даних використовується клас QDataWidgetMapper. Редагування проводиться таким чином: методом setMapping(віджет, стовпець) вказується, які поля в формі редагування запису потрібно зв'язати зі стовпцями в таблиці, при виклику методу submit(), відредаговані дані відправляються в БД.

Для роботи з класом EditCustomer потрібні були такі класи:

- QDataWidgetMapper - для зв'язку віджетів зі стовпцями таблиці,
- QMessageBox - для виведення інформації про некоректне введення.

Клас успадкований від QDialog.

Клас AddMoney

Даний клас потрібен для редагування фінансової складової облікового запису клієнта, реалізовані наступні дії:

- підключення клієнта,
- відключення (наприклад, за власним бажанням),
- відключення за борги (при відключенні за борги баланс змінюється на -1),

- редагування балансу.

При редагуванні балансу, дата редагування буде відправлена в стовпець `LAST_PAY` в базі даних, а зміна статусу підключення буде відправлена в стовпець `STATUS`. Редагування здійснюється за допомогою того ж `QDataWidgetMapper`.

Для роботи з класом `AddMoney` потрібні були такі класи:

- `QDataWidgetMapper` - для зв'язку віджетів зі стовпцями таблиці,
- `QDate` - для введення дати останнього платежу,
- `QTimer` - для відображення поточного часу,
- `QMessageBox` - для виведення інформації про некоректне введення.

Клас успадкований від `QDialog`.

Клас `OutputDocuments`

Даний клас призначений для виведення інформації з бази даних в текстові файли. Підтримується виведення такої інформації:

- історія платежів,
- список боржників,
- сортування (виведення клієнтів з певного міста),
- прайс-лист на послуги.

Для отримання даних з БД, методи цього класу звертаються до методів класу `Database`, який у свою чергу отримує запрошену інформацію з бази даних. Інформація з бази даних поступає в цей клас у вигляді контейнера `QList`, що містить дані типу `QString`. Дані з контейнера виводяться в віджет `QListWidget`, є кнопка, при натисненні на яку відкривається діалог збереження файлу і дані записуються у вказаний файл. Для сортування викликається запит `SQL WHERE CITY = 'місто'`, запит повертає облікові

записи усіх користувачів із запрошеного міста. У програмі реалізовано 3 міста Донецької області:

- Маріуполь,
- Покровськ,
- Авдіївка.

Для роботи з класом `OutputDocuments` потрібні були такі класи:

- `QFile` - запис в файл
- `QFileDialog` - запрошення зберегти файл
- `Database` - робота з базою даних.

Клас успадкований від `QDialog`.

Клас `MainWindow`

Даний клас є командним центром програми, містить в собі табличне представлення і кнопки:

- додати,
- редагувати,
- керування балансом,
- видалити,
- вихідні документи,
- очистити таблицю.

У конструкторі даного класу реалізовано відкриття БД. Спочатку перевіряється наявність файлу `customers.sqlite` в директорії з виконуваним файлом, в разі, якщо він існує, то виконується метод `connectToDatabase ()`, класу `Database`, якщо файлу не існує, то за допомогою статичного методу `getOpenFileName` класу `QFileDialog` пропонується вказати шлях до бази даних, після вказівки, викликається перевантажений метод `connectToDatabase`, що приймає в якості аргументу рядок, в якому міститься шлях до файлу БД, отриманий з попереднього кроку. Читання записів з бази

даних проводиться через клас QSqlTableModel[4], об'єкт цього класу - модель табличного представлення використовується в компоненті QTableView на головній формі програми. Варто звернути увагу на те, що в таблиці на головній формі присутні 13 стовпців, але в БД їх 14. З міркувань безпеки був прихований стовпець ID, який є первинним ключем. Він потрібен для того, щоб можна було однозначно ідентифікувати необхідний запис в БД, проте, якщо його пошкодити, або ввести некоректне значення, то БД може перестати коректно функціонувати. У класі присутні методи для оновлення таблиці:

- розтягування стовпців, згідно їх вмісту,
- оновлення записів в таблиці,
- очищення таблиці.

Також реалізований пошук за прізвищем клієнта.

Для роботи з класом MainWindow потрібні були такі класи:

- QSqlTableModel - надає модель табличного представлення,
- Database - робота з БД,
- QFile - отримання шляху до БД,
- QFileDialog - запрошення вказати шлях до БД,
- QMessageBox - інформаційні повідомлення,
- EditCustomer - редагування запису,
- AddCustomer - додавання запису,
- AddMoney - фінансові операції,
- OutputDocuments - вихідні документи.

2.2 Формалізація опису класів у вигляді діаграми класів

Діаграма класів була оформлена з використанням нотацій мови UML. Діаграма класів з коментарями до неї є результатом даного етапу роботи і

розміщується в додатку А до ПЗ. У таблиці 1 наведено назву класу, назву батьківського класу, та короткий опис.

Таблиця 1 – опис класів, які застосовувалися у курсовій роботі

Назва класу	Батьківський клас	Опис
AddCustomer	QDialog	Додавання нового запису до БД
AddMoney	QDialog	Фінансові операції
Database	QObject	Робота з БД
EditCustomer	QDialog	Редагування запису
MainWindow	QMainWindow	Головне вікно
OutputDocuments	QDialog	Вихідні документи

2.3 Програмна реалізація класів

Повний вихідний програмний код із визначеннями методів класів розміщений у додатку Б до пояснювальної записки. Слід звернути увагу, що тут мова йде тільки про програмний код, що реалізує систему класів предметної області. Розглянемо методи та їх атрибути.

Клас AddCustomer:

```
private slots:
    void addCustomer ()
    void check ()
    void closeWindow ()
signals:
    void applyChanges ()
```

Клас AddMoney:

```
public:
    void setModel (QAbstractItemModel *model)
private slots:
    void addMoney ()
    void updateTime ()
    void compare ()

signals:
    void applyChanges ()
```

Клас Database:

```
void connectToDatabase();
void connectToDatabase(QString &filePath)
void insertIntoTable(QVariantList &data)
void dropTable()
QList<QString> getPaymentsHistory()
QList<QString> getDebtors()
QList<QString> getCities(QString city)
```

Клас EditCustomer:

```
public:
    void setModel(QAbstractItemModel *model)
private slots:
    void editCustomer()
    void check()
signals:
    void applyChanges()
```

Клас MainWindow:

```
private slots:
    void search(const QString &searchValue)
    void addCustomer()
    void removeCustomer()
    void editCustomer()
    void addMoney()
    void resizeTable()
    void updateTable()
    void clearTable()
    void contextMenu(QPoint pos)
    void selectRow()
signals:
    void customerRemoved()
private:
    void setupModel()
    void openDatabase()
    void setupWindow()
    void setConnects()
```

Клас OutputDocuments:

```
private slots:
    void getPaymentsHistory()
    void getDebtorsList()
    void getSortedList(int city)
    void savePaymentsHistory()
    void saveDebtorsList()
    void saveSortedList()
    void savePriceList()
    void check(int index)
    void update()
```

3. РОЗРОБКА ІНТЕРФЕЙСУ ПРОГРАМИ

3.1 Опис користувацького інтерфейсу

Запуск програми

Як вже було сказано вище, програма спочатку шукає файл бази даних у директорії з виконуючим файлом, якщо його там немає, програма просить вказати шлях до бази даних (рис. 3.1, 3.2).

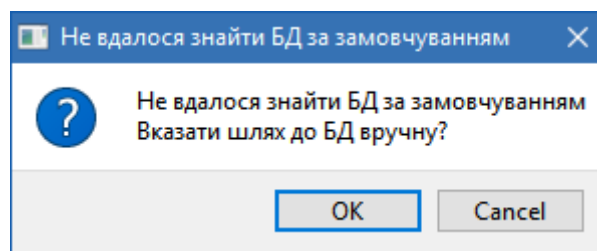


Рисунок 3.1 – Запрошення вказати шлях до бази даних

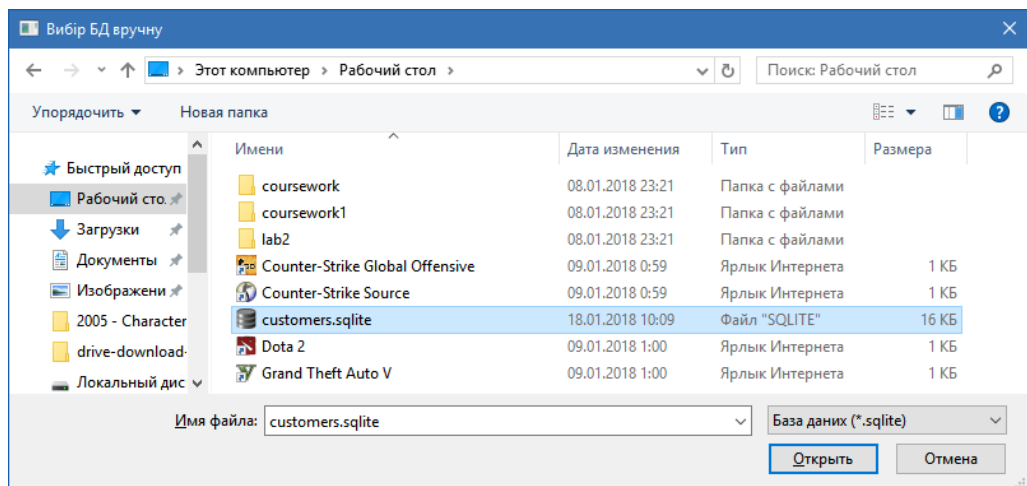
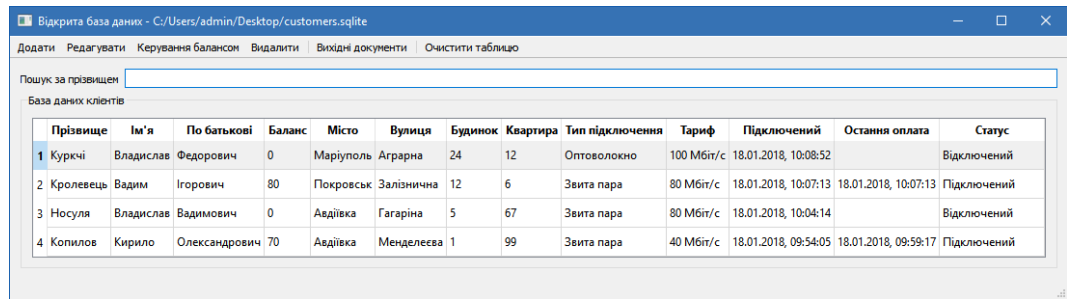


Рисунок 3.2 – Діалог вибору файлу



Відкрита база даних - C:/Users/admin/Desktop/customers.sqlite

Додати Редагувати Керування балансом Видалити Вихідні документи Очистити таблицю

Пошук за прізвищем

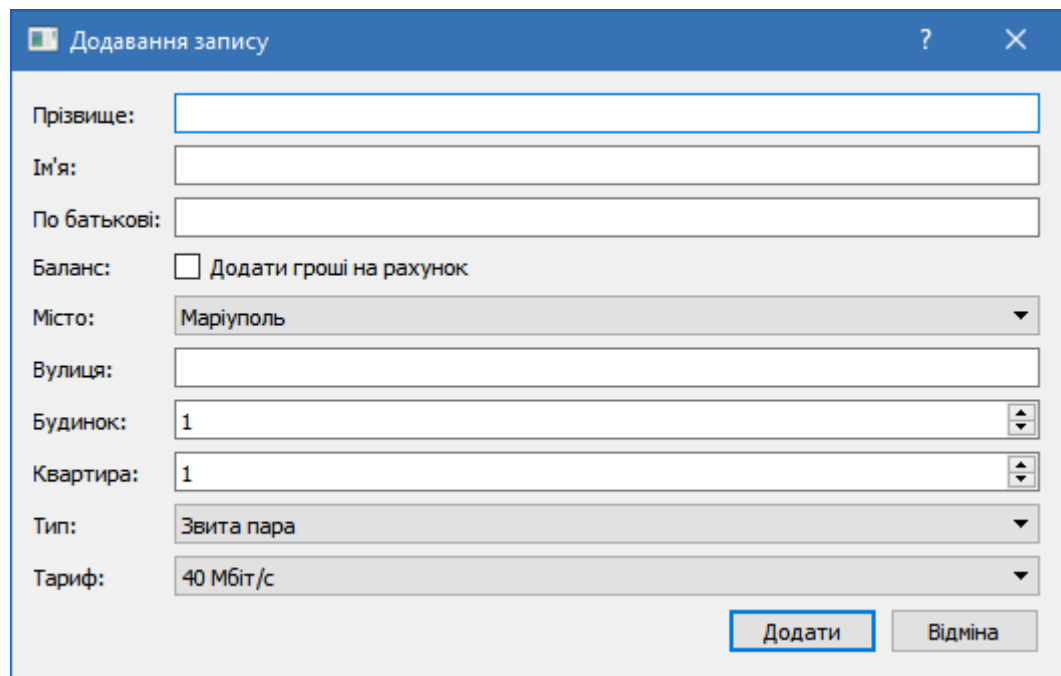
База даних клієнтів

	Прізвище	Ім'я	По батькові	Баланс	Місто	Вулиця	Будинок	Квартира	Тип підключення	Тариф	Підключений	Остання оплата	Статус
1	Курччі	Владислав	Федорович	0	Маріуполь	Аграрна	24	12	Оптоволокно	100 Мбіт/с	18.01.2018, 10:08:52		Відключений
2	Кролевець	Вадим	Ігорович	80	Покровськ	Залізнична	12	6	Звита пара	80 Мбіт/с	18.01.2018, 10:07:13	18.01.2018, 10:07:13	Підключений
3	Носуля	Владислав	Вадимович	0	Авдіївка	Гагаріна	5	67	Звита пара	80 Мбіт/с	18.01.2018, 10:04:14		Відключений
4	Копилов	Кирило	Олександрович	70	Авдіївка	Менделєєва	1	99	Звита пара	40 Мбіт/с	18.01.2018, 09:54:05	18.01.2018, 09:59:17	Підключений

Рисунок 3.3 – Головне вікно програми

Розглянемо додавання нового запису до бази даних

Натискаємо кнопку Додати у головному вікні програми, відкриється форма, зображена на рисунку 3.4.



Додавання запису

Прізвище:

Ім'я:

По батькові:

Баланс: ☐ Додати гроші на рахунок

Місто:

Вулиця:

Будинок:

Квартира:

Тип:

Тариф:

Рисунок 3.4 – Форма додавання запису до бази даних

Щоб додати новий запис, обов'язково треба заповнити усі поля для вводу, якщо цього не зробити, то програма видасть попередження (рис. 3.5).

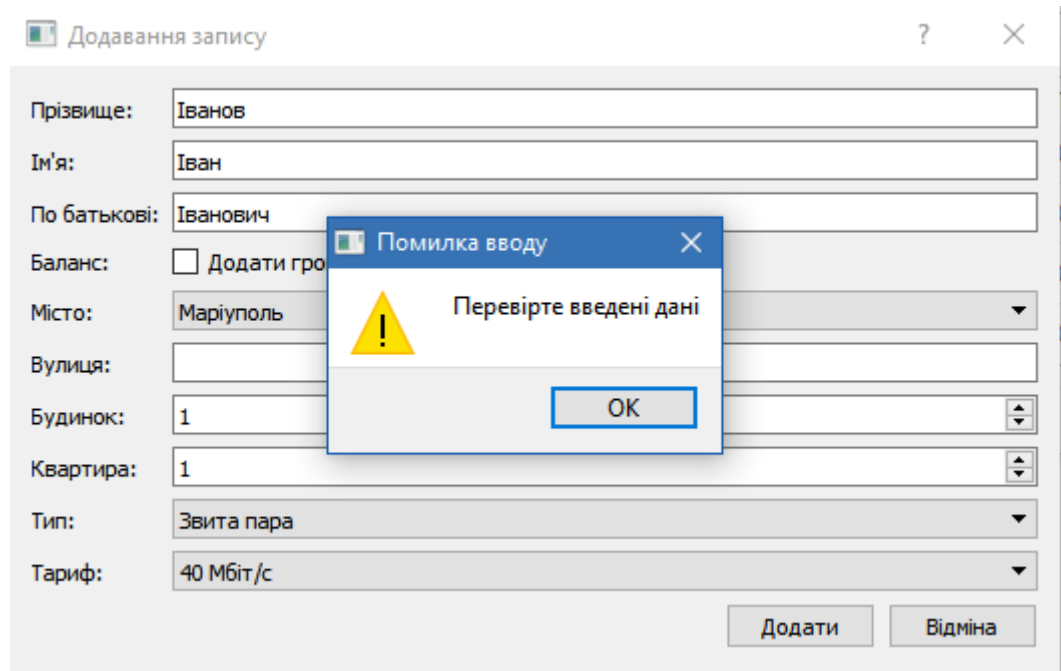


Рисунок 3.5 – Перевірка коректності даних

Після успішної перевірки введених даних, новий запис буде додано до бази даних.

Розглянемо редагування запису

Натискаємо кнопку Редагувати у головному вікні програми, відкриється форма, зображена на рисунку 3.6.

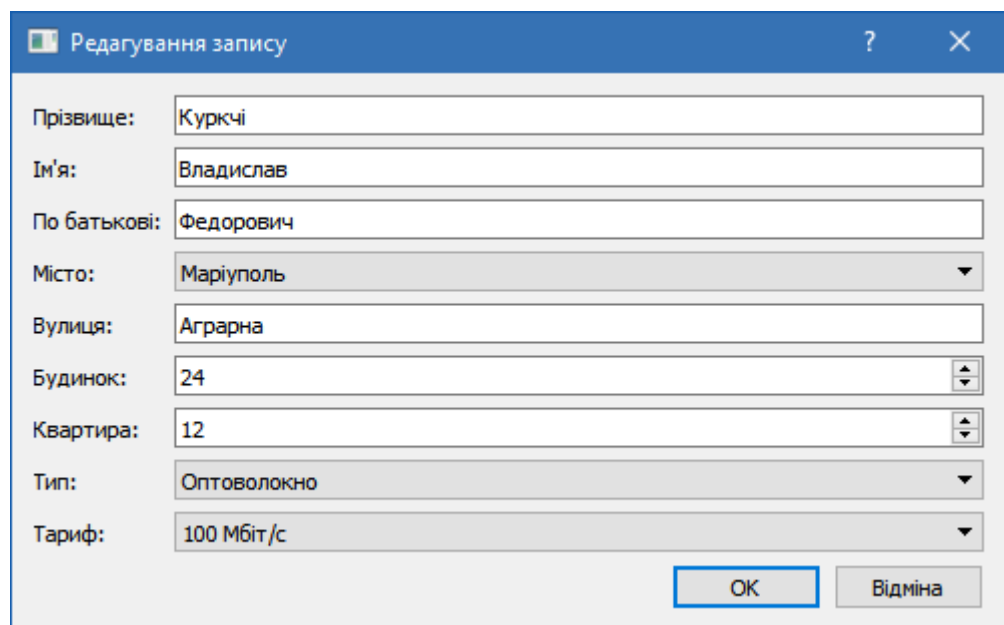


Рисунок 3.6 – Форма редагування запису

Рисунок 3.7 – Перевірка коректності даних

Розглянемо фінансові операції

Натискаємо кнопку Керування балансом у головному вікні програми, відкриється форма, зображена на рисунку 3.8.

Рисунок 3.8 – Фінансові операції з обліковим записом

Рисунок 3.9 – Підключення облікового запису

Відкрита база даних - C:/Users/admin/Desktop/customers.sqlite

Додати Редагувати Керування балансом Видалити Вихідні документи Очистити таблицю

Пошук за прізвищем

База даних клієнтів

	Прізвище	Ім'я	По батькові	Баланс	Місто	Вулиця	Будинок	Квартира	Тип підключення	Тариф	Підключений	Остання оплата	Статус
1	Куркчі	Владислав	Федорович	124	Маріуполь	Аграрна	24	12	Оптоволокно	100 Мбіт/с	18.01.2018, 10:08:52	18.01.2018, 11:00:31	Підключений
2	Кролевець	Вадим	Ігорович	80	Покровськ	Залізнична	12	6	Звита пара	80 Мбіт/с	18.01.2018, 10:07:13	18.01.2018, 10:07:13	Підключений
3	Носуля	Владислав	Вадимович	0	Авдіївка	Гагаріна	5	67	Звита пара	80 Мбіт/с	18.01.2018, 10:04:14		Відключений
4	Копилов	Кирило	Олександрович	70	Авдіївка	Менделєєва	1	99	Звита пара	40 Мбіт/с	18.01.2018, 09:54:05	18.01.2018, 09:59:17	Підключений

Рисунок 3.10 – Після внесення грошей на баланс, статус облікового запису змінився на «Підключений», також змінилася «Остання оплата»

Розглянемо формування вихідних документів

Натискаємо кнопку Вихідні документи у головному вікні програми, відкриється форма, зображена на рисунку 3.10.

Вихідні документи

Історія платежів Список боржників Сортування Прайс-лист

18.01.2018, 09:59:17 -> Копилов Кирило Олександрович
 18.01.2018, 10:07:13 -> Кролевець Вадим Ігорович
 18.01.2018, 11:02:19 -> Куркчі Владислав Федорович

Зберегти в файл

Оновити

Рисунок 3.10 – Форма формування вихідних документів, історія платежів

Для збереження в файл, натискаємо кнопку «Зберегти в файл», відкриється запрошення вказати куди саме зберігати файл (рисунок 3.11).

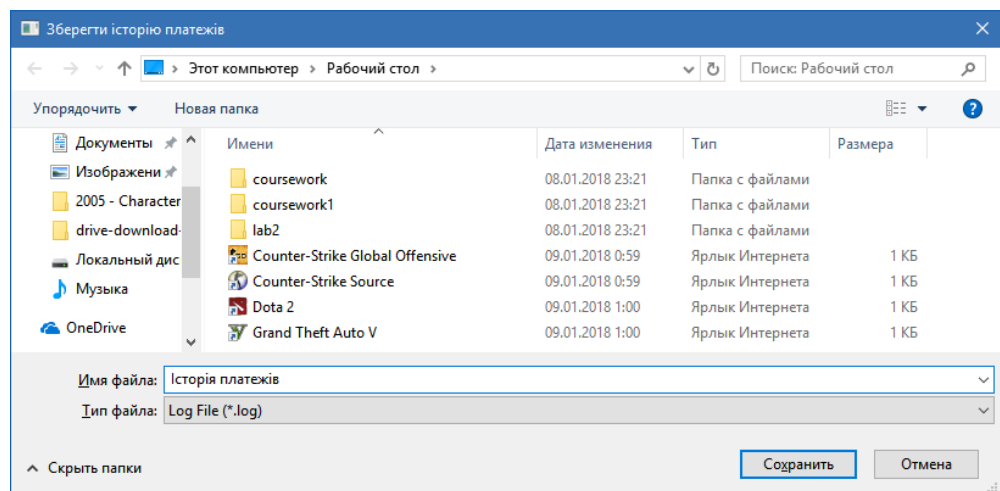


Рисунок 3.11 – Запрошення зберегти файл

На рисунку 3.12 розглянемо зміст збереженого файлу.

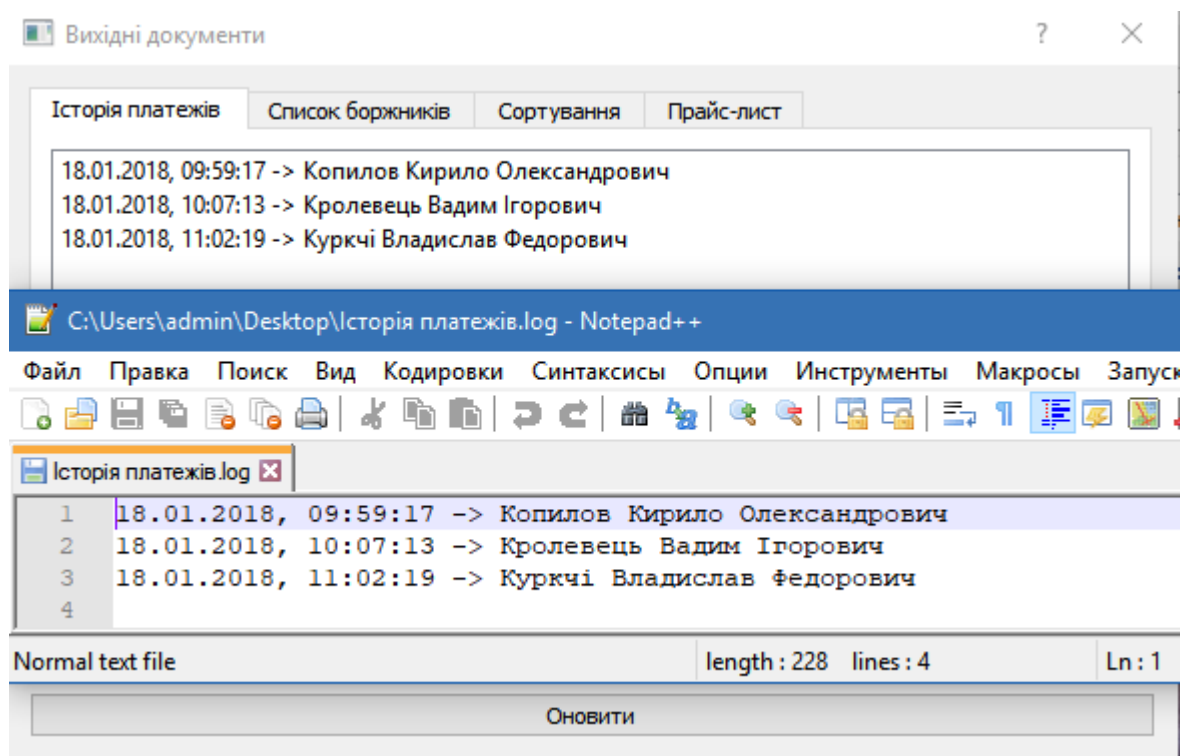


Рисунок 3.12 – Зміст збереженого файлу

Розглянемо збереження, та зміст списку боржників.

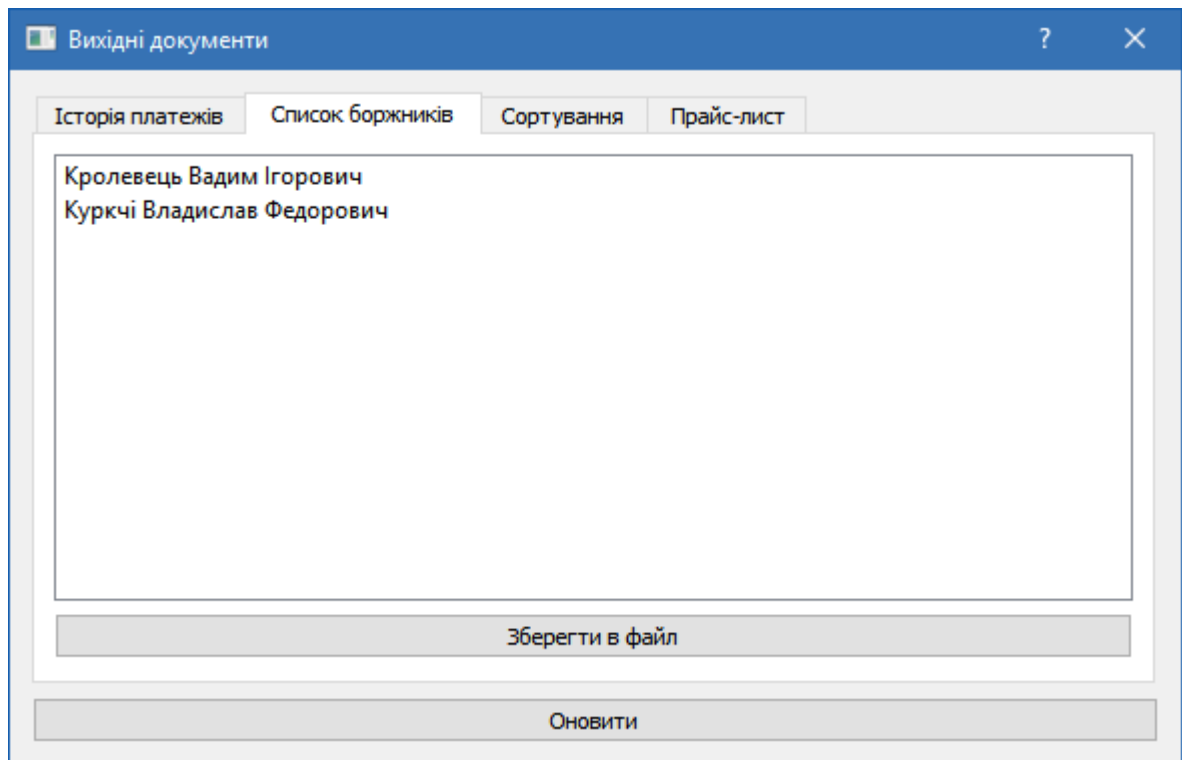


Рисунок 3.13 – Список боржників

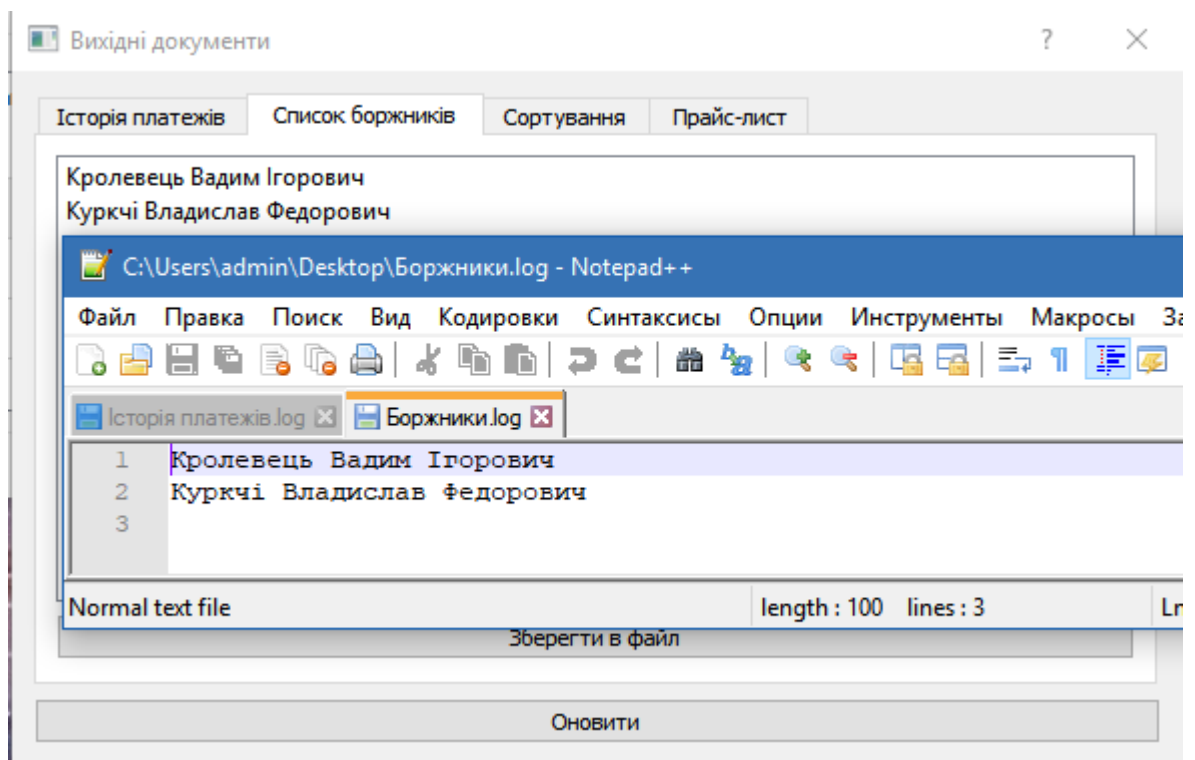


Рисунок 3.14 – Зміст файлу, у який був збережений список боржників

Розглянемо формування списку за містом.

Переходимо на вкладку «Сортування» (рис 3.15).

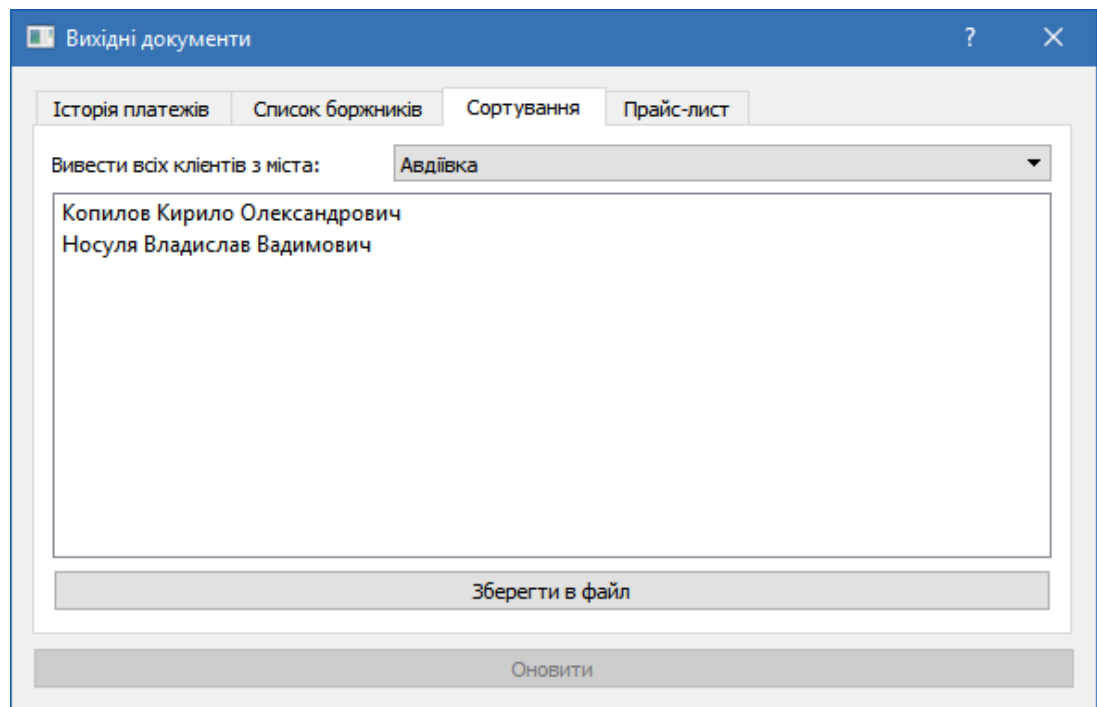


Рисунок 3.15 – Вкладка Сортування

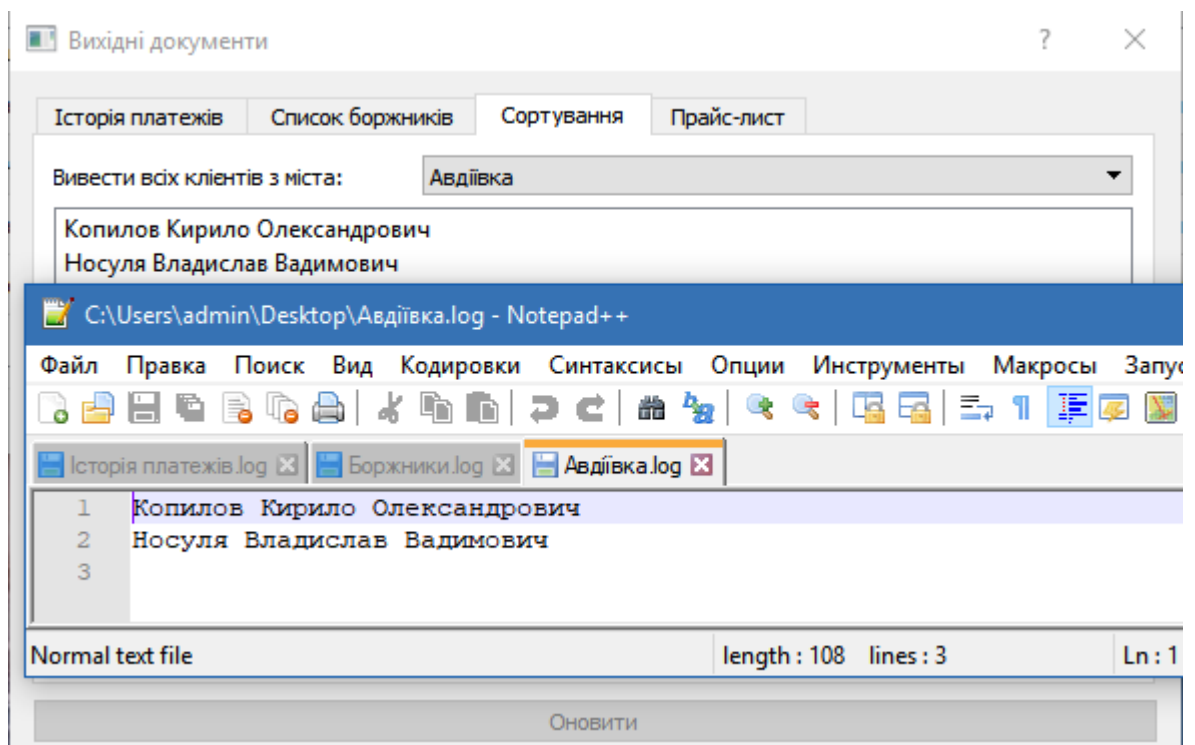


Рисунок 3.16 – Збережемо у файл клієнтів, наприклад з міста Авдіївка

Розглянемо формування прайс-листа

Перейдемо на вкладку «Прайс-лист», та заповнимо поля вводу (рисунок 3.17).

Вихідні документи

Історія платежів Список боржників Сортуння **Прайс-лист**

Вартість підключення (багатоповерхівки): 1

Вартість підключення (приватний сектор): 2

Виклик фахівця на об'єкт: 3

Метр звітої пари: 4

Налаштування ОС: 5

Налаштування обладнання: 6

Зберегти в файл

Оновити

Рисунок 3.17 – Форма формування прайс-листа

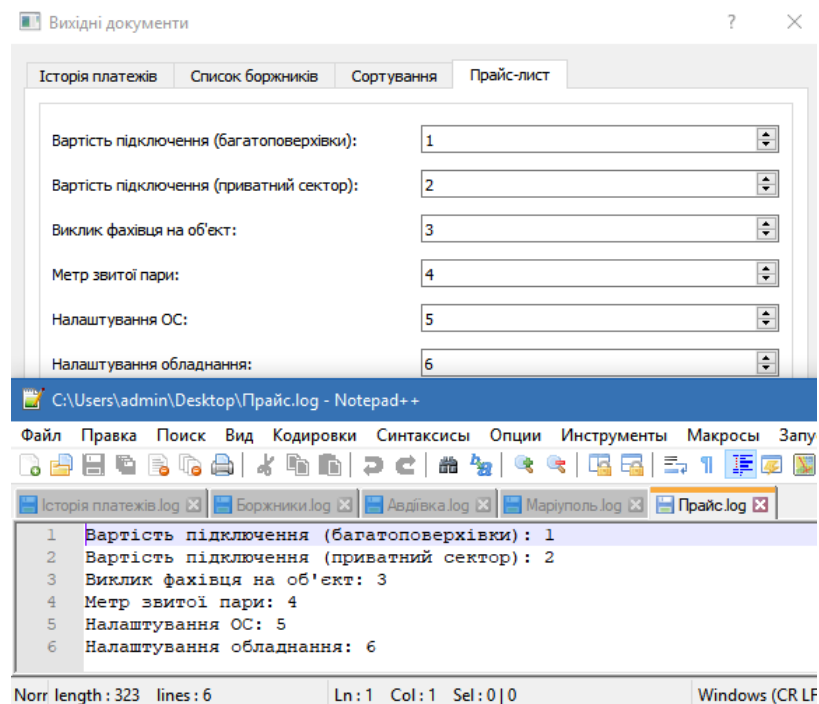


Рисунок 3.18 – Зміст збереженого файлу прайс-листа

Видалення та очищення таблиці не є чимось складним, розглянемо видалення запису та очистку таблиці.

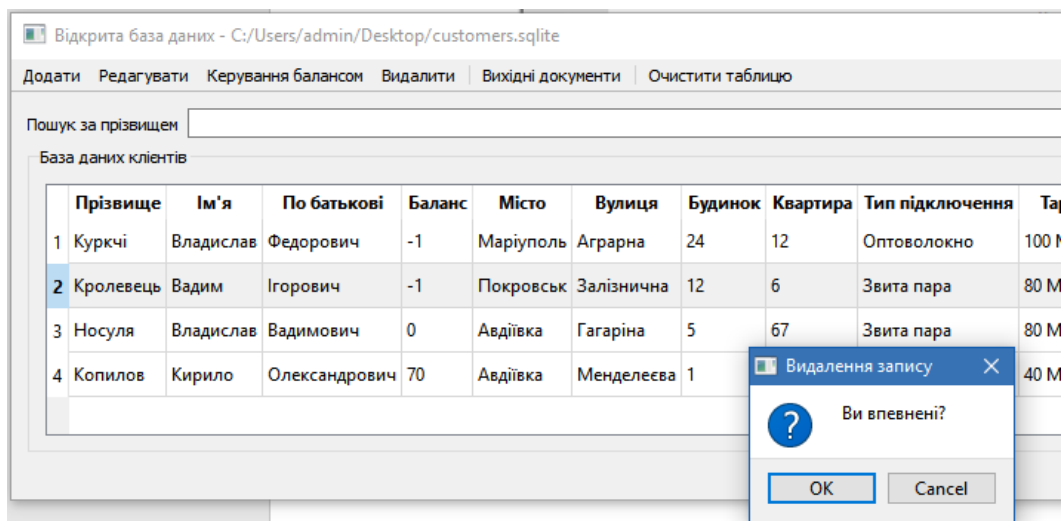


Рисунок 3.19 – Попередження про видалення запису

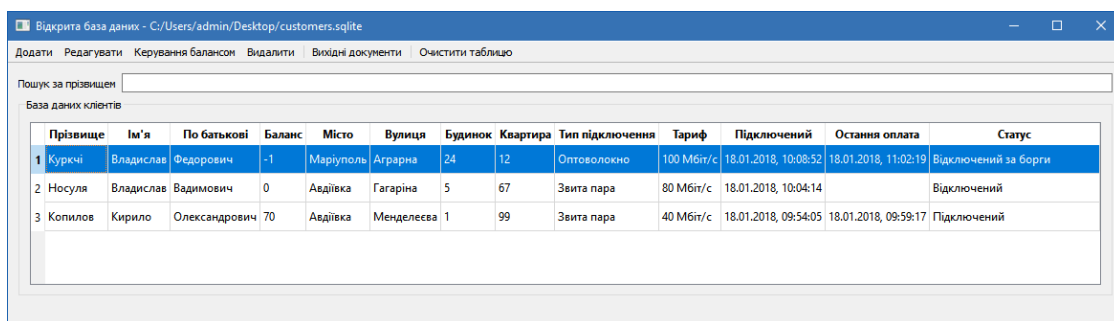


Рисунок 3.20 – Таблиця після видалення одного запису

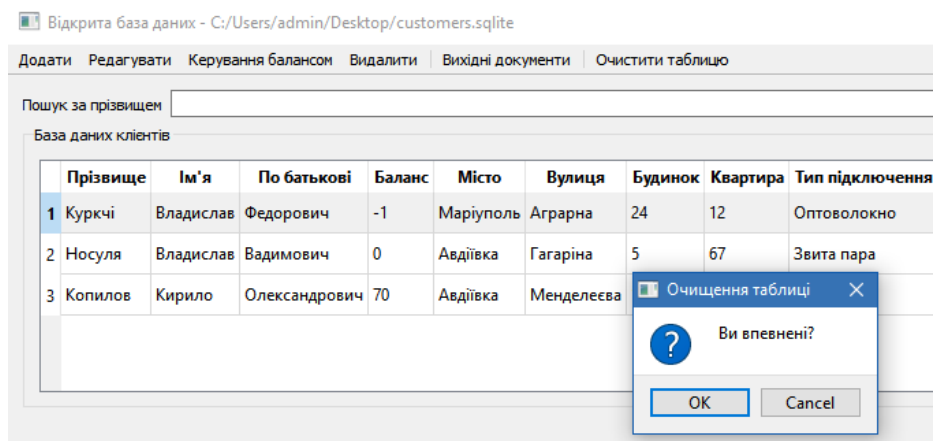


Рисунок 3.21 – Попередження про очищення таблиці

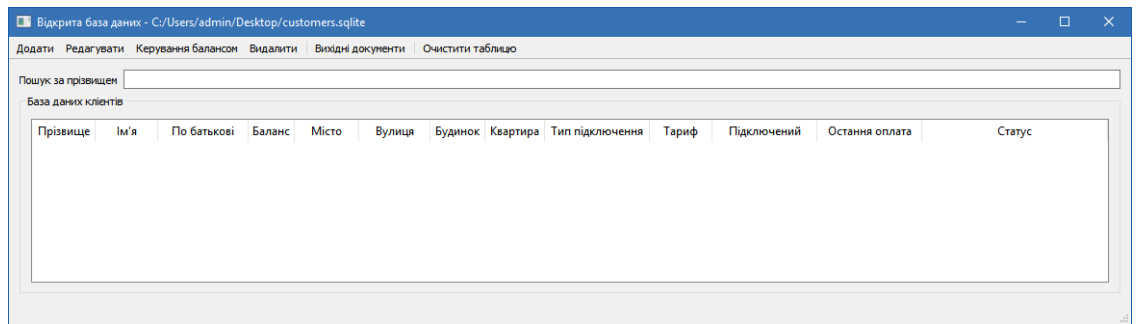


Рисунок 3.22 – Очищена таблиця

3.2 Програмна реалізація інтерфейсу

Програма використовує графічний інтерфейс користувача – Qt Widgets. Віджети - це вихідні елементи для створення призначеного для користувача інтерфейсу в Qt. Віджети можуть відображати дані та інформацію про стан, отримати введення від користувача і надавати контейнер для інших віджетів, які повинні бути згруповані. Віджет, що не вбудований в батьківський віджет, називається вікном [5].

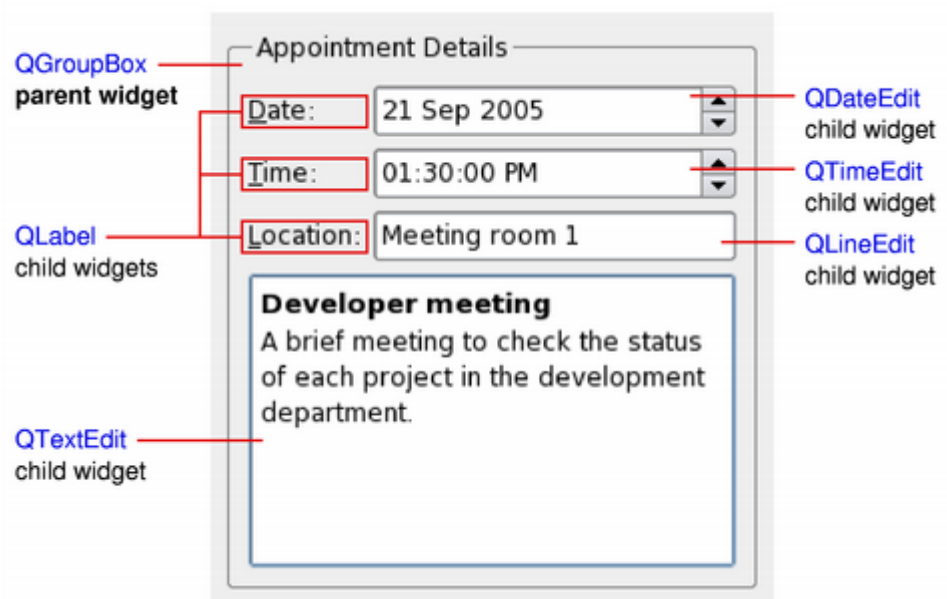


Рисунок 3.2.1 – приклад віджета в Qt

Повний лістинг реалізації інтерфейсу є у додатку В до ПЗ.

ВИСНОВКИ

Виконавши курсову роботу, я познайомився з парадигмою ООП, отримав досвід у розробці прикладного ПО, познайомився з фреймворком Qt, механізмом сигналів та слотів, мовою SQL.

Що було виконано:

- додавання даних до бази даних,
- редагування даних у базі даних,
- видалення даних з бази даних,
- виведення даних з бази даних до файлу.

Що не було виконано:

- створення нової бази даних,
- створення нової таблиці,
- захист від іншої бази даних.

Що можна поліпшити, або додати:

- множинне видалення,
- механізм регулярних виразів,
- створення нової бази даних,
- реєстронезалежний пошук,
- меню налаштувань.

СПИСОК ЛІТЕРАТУРИ

1. Об'єктно-орієнтоване програмування [Електронний ресурс] – Режим доступу:

https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

2. База даних SQLite [Електронний ресурс] – Режим доступу:

<https://uk.wikipedia.org/wiki/SQLite>

3. Механізм сигналів та слотів [Електронний ресурс] – Режим доступу:

<http://doc.crossplatform.ru/qt/4.3.2/signalsandslots.html>

4. Клас QSqlTableModel [Електронний ресурс] – Режим доступу:

<https://evileg.com/ru/post/62/>

5. Віджети у Qt [Електронний ресурс] – Режим доступу:

<http://doc.crossplatform.ru/qt/4.7.x/widgets-and-layouts.html>

ДОДАТОК А – ДІАГРАМА КЛАСІВ

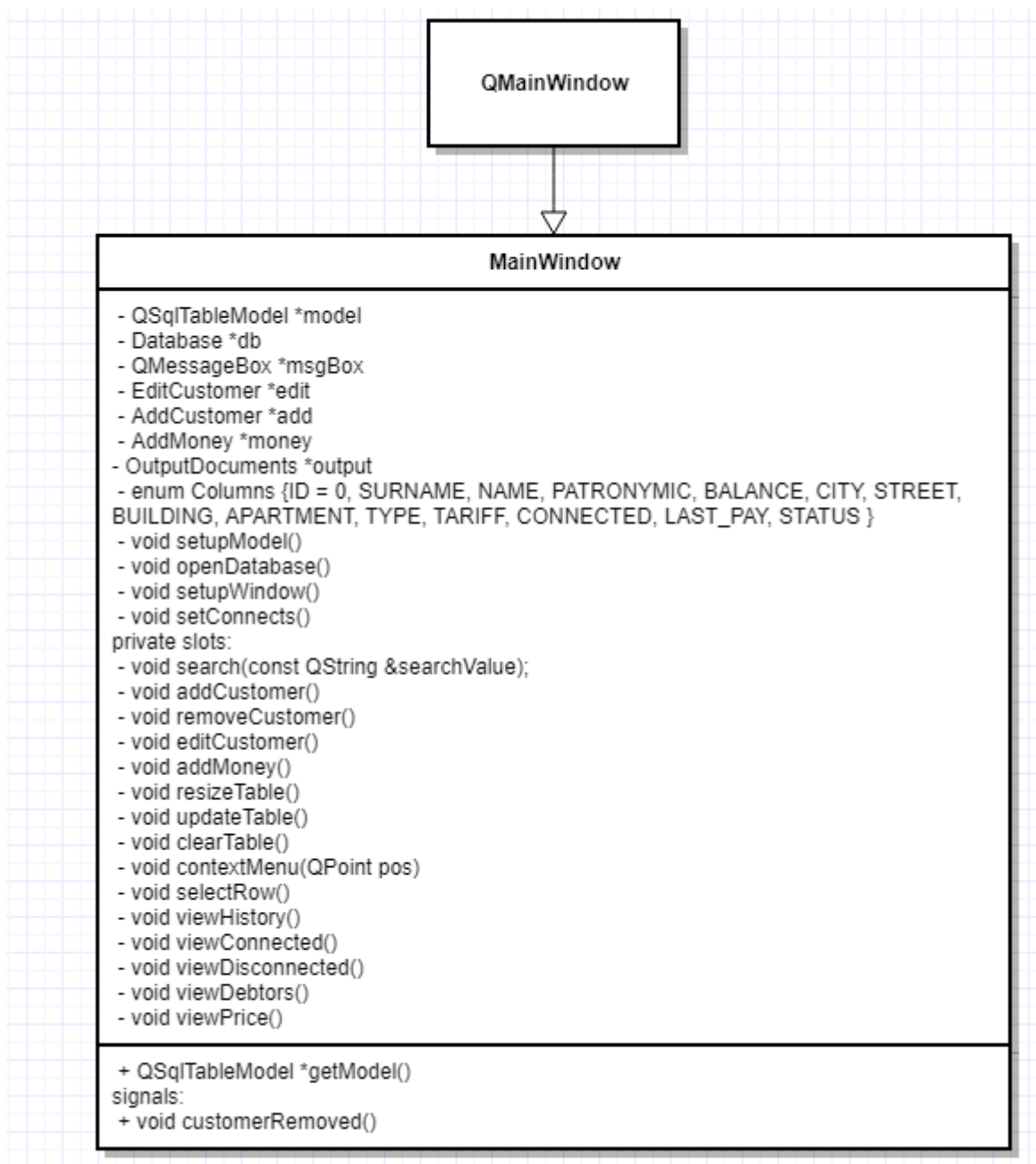


Рисунок 1 – Діаграма класу MainWindow

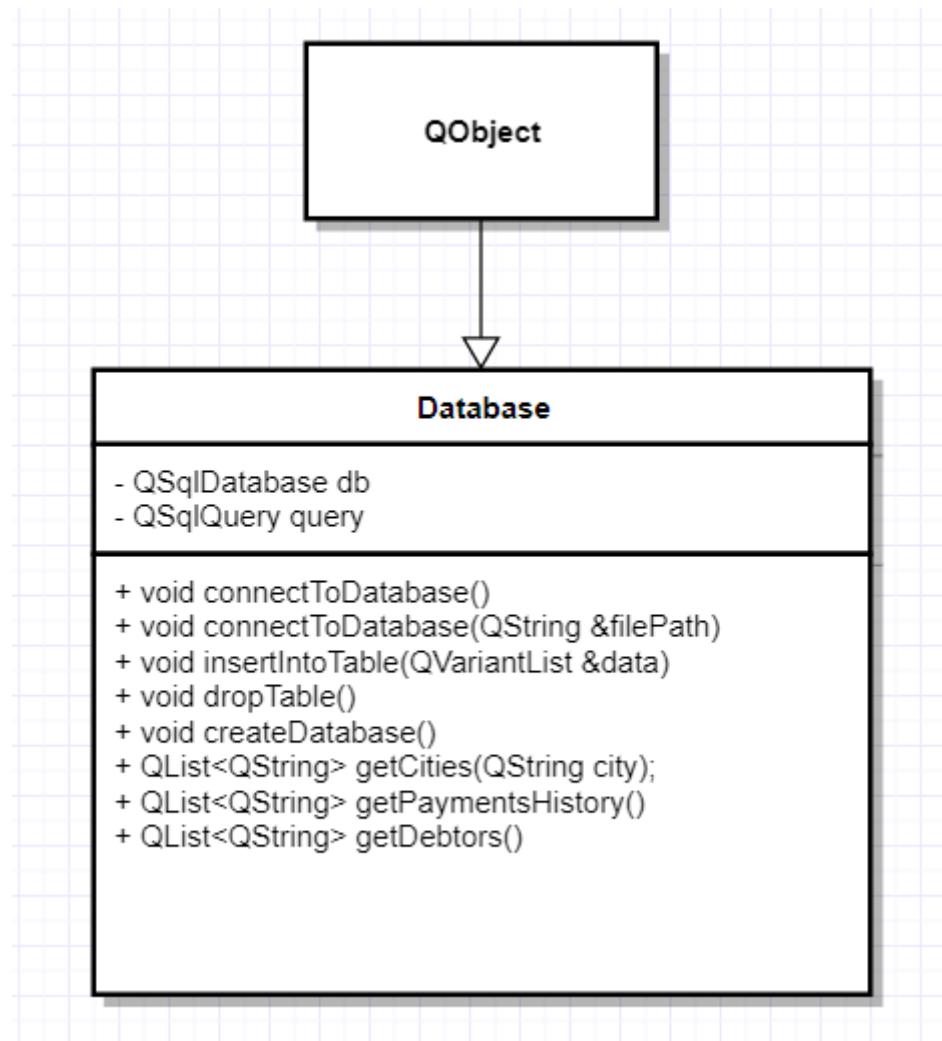


Рисунок 2 – Діаграма класу Database

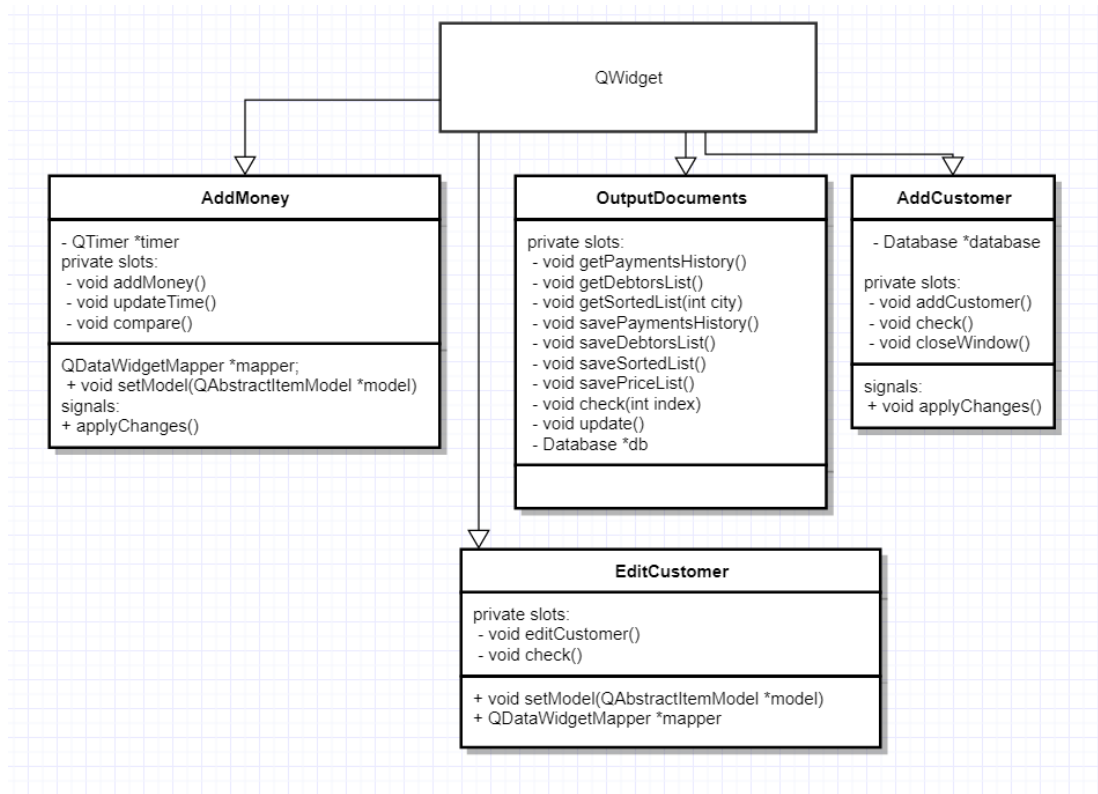


Рисунок 3 – Діаграми класів AddMoney, OutputDocuments, AddCustomer, EditCustomer

ДОДАТОК Б - ЛІСТИНГ РЕАЛІЗАЦІЇ КЛАСІВ

addcustomer.h

```

#ifndef ADDCUSTOMER_H
#define ADDCUSTOMER_H

#include <QDialog>
#include <QDate>
#include <QMessageBox>
#include "database.h"

namespace Ui {
class AddCustomer;
}

class AddCustomer : public QDialog
{
    Q_OBJECT

public:
    explicit AddCustomer(QWidget *parent = 0);
    ~AddCustomer();

private slots:
    void addCustomer();
    void check();
    void closeWindow();
signals:
    void applyChanges();

private:
    Ui::AddCustomer *ui;
    Database *database;
};

#endif // ADDCUSTOMER_H

```

addmoney.h

```

#ifndef ADDMONEY_H
#define ADDMONEY_H

#include <QDialog>
#include <QDataWidgetMapper>
#include <QDate>
#include <QTimer>
#include <QMessageBox>
#include "database.h"

namespace Ui {
class AddMoney;
}

class AddMoney : public QDialog
{
    Q_OBJECT

public:
    explicit AddMoney(QWidget *parent = 0);
    ~AddMoney();
    QDataWidgetMapper *mapper;
    void setModel(QAbstractItemModel *model);
private slots:
    void addMoney();
    void updateTime();
    void compare();
signals:
    void applyChanges();

private:

```

```

        Ui::AddMoney *ui;
        QTimer *timer;
};

#endif // ADDMONEY_H

```

database.h

```

#ifndef DATABASE_H
#define DATABASE_H

#include <QObject>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QDebug>
#include <QSqlError>
#include <QSqlRecord>
#include <QApplication>

class Database : public QObject
{
    Q_OBJECT
public:
    explicit Database(QObject *parent = nullptr);
    void connectToDatabase();
    void connectToDatabase(QString &filePath);
    void insertIntoTable(QVariantList &data);
    void dropTable();
    QList<QString> getPaymentsHistory();
    QList<QString> getDebtors();
    QList<QString> getCities(QString city);

private:
    QSqlDatabase db;
    QSqlQuery query;
};

#endif // DATABASE_H

```

editcustomer.h

```

#ifndef EDITCUSTOMER_H
#define EDITCUSTOMER_H

#include <QDialog>
#include <QDataWidgetMapper>
#include <QMessageBox>

namespace Ui {
class EditCustomer;
}

class EditCustomer : public QDialog
{
    Q_OBJECT

public:
    explicit EditCustomer(QWidget *parent = 0);
    ~EditCustomer();
    void setModel(QAbstractItemModel *model);
    QDataWidgetMapper *mapper;
private slots:
    void editCustomer();
    void check();
signals:
    void applyChanges();
private:
    Ui::EditCustomer *ui;
};

#endif // EDITCUSTOMER_H

```


mainwindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QSqlTableModel>
#include "database.h"
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include "editcustomer.h"
#include "addcustomer.h"
#include "addmoney.h"
#include "outputdocuments.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    QSqlTableModel *getModel();

private slots:
    void search(const QString &searchValue);
    void addCustomer();
    void removeCustomer();
    void editCustomer();
    void addMoney();
    void resizeTable();
    void updateTable();
    void clearTable();
    void contextMenu(QPoint pos);
    void selectRow();

signals:
    void customerRemoved();

private:
    Ui::MainWindow *ui;
    QSqlTableModel *model;
    Database *db;
    QMessageBox *msgBox;
    EditCustomer *edit;
    AddCustomer *add;
    AddMoney *money;
    OutputDocuments *output;
    enum Columns {
        ID = 0,
        SURNAME,
        NAME,
        PATRONYMIC,
        BALANCE,
        CITY,
        STREET,
        BUILDING,
        APARTMENT,
        TYPE,
        TARIFF,
        CONNECTED,
        LAST_PAY,
        STATUS
    };
    void setupModel();
    void openDatabase();
    void setupWindow();
    void setConnects();

```

```
};

#endif // MAINWINDOW_H
```

outputdocuments.h

```
#ifndef OUTPUTDOCUMENTS_H
#define OUTPUTDOCUMENTS_H

#include <QDialog>
#include <QFile>
#include <QFileDialog>
#include "database.h"

namespace Ui {
class OutputDocuments;
}

class OutputDocuments : public QDialog
{
    Q_OBJECT

public:
    explicit OutputDocuments(QWidget *parent = 0);
    ~OutputDocuments();
private slots:
    void getPaymentsHistory();
    void getDebtorsList();
    void getSortedList(int city);
    void savePaymentsHistory();
    void saveDebtorsList();
    void saveSortedList();
    void savePriceList();
    void check(int index);
    void update();
private:
    Ui::OutputDocuments *ui;
    Database *db;
};

#endif // OUTPUTDOCUMENTS_H
```

addcustomer.cpp

```
#include "addcustomer.h"
#include "ui_addcustomer.h"

AddCustomer::AddCustomer(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AddCustomer)
{
    ui->setupUi(this);
    database = new Database(this);
    connect(ui->pbApply, SIGNAL(clicked(bool)), this, SLOT(check()));
    connect(ui->pbCancel, SIGNAL(clicked(bool)), this, SLOT(closeWindow()));
    connect(ui->cbMoney, SIGNAL(clicked(bool)), ui->spBalance,
    SLOT(setVisible(bool)));
    ui->spBalance->setVisible(false);
    QRegExp expression("[A-ЯІЄ{1}][a-яґєії]{39}");
    QRegExpValidator *validator = new QRegExpValidator(expression, this);
    ui->lineName->setValidator(validator);
    ui->lineSurname->setValidator(validator);
    ui->linePatronymic->setValidator(validator);
    ui->lineStreet->setValidator(validator);
}

AddCustomer::~AddCustomer()
{
    delete ui;
}
```

```

void AddCustomer::addCustomer()
{
    QString name = ui->lineName->text();
    QString surname = ui->lineSurname->text();
    QString patronymic = ui->linePatronymic->text();
    int balance = ui->spBalance->text().toInt();
    QString city = ui->cbCities->currentText();
    QString street = ui->lineStreet->text();
    int building = ui->spBuilding->text().toInt();
    int apartment = ui->spApartment->text().toInt();
    QString type = ui->cbConnectionTypes->currentText();
    QString tariff = ui->cbTariffs->currentText();
    QString status;
    QString connected = QDateTime::currentDateTime().toString("dd.MM.yyyy,
hh:mm:ss");
    QString last_pay;
    if(ui->cbMoney->isChecked() && ui->spBalance->value() > 0) {
        last_pay = connected;
        status = "Підключений";
    } else {
        last_pay = "";
        status = "Відключений";
    }
    foreach(QLineEdit *le, findChildren<QLineEdit*>()) {
        le->clear();
    }
    ui->cbMoney->setChecked(false);
    ui->spBalance->setVisible(false);
    ui->spBalance->setValue(0);
    QVariantList data;
    data.append(surname);
    data.append(name);
    data.append(patronymic);
    data.append(balance);
    data.append(city);
    data.append(street);
    data.append(building);
    data.append(apartment);
    data.append(type);
    data.append(tariff);
    data.append(connected);
    data.append(last_pay);
    data.append(status);
    database->insertIntoTable(data);
    emit applyChanges();
    close();
}

void AddCustomer::check()
{
    if(ui->lineName->text().length() < 2
        || ui->lineSurname->text().length() < 2
        || ui->linePatronymic->text().length() < 2
        || ui->lineStreet->text().length() < 2)
        QMessageBox::warning(this, "Помилка вводу", "Перевірте введені дані");
    else
        this->addCustomer();
}

void AddCustomer::closeWindow()
{
    foreach(QLineEdit *le, findChildren<QLineEdit*>()) {
        le->clear();
    }
    close();
}

```

addmoney.cpp

```

#include "addmoney.h"
#include "ui_addmoney.h"

```

```

AddMoney::AddMoney(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AddMoney)
{
    ui->setupUi(this);
    mapper = new QDataWidgetMapper(this);
    timer = new QTimer(this);
    mapper->setSubmitPolicy(QDataWidgetMapper::ManualSubmit);
    connect(ui->pbApply, SIGNAL(clicked(bool)), this, SLOT(addMoney()));
    connect(ui->pbCancel, SIGNAL(clicked(bool)), this, SLOT(close()));
    connect(ui->cbConnectionStatus, SIGNAL(currentIndexChanged(int)), this,
    SLOT(compare()));
    connect(timer, SIGNAL(timeout()), this, SLOT(updateTime()));
    timer->start(1000);
}

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

void AddMoney::setModel(QAbstractItemModel *model)
{
    mapper->setModel(model);
    mapper->addMapping(ui->spBalance, 4);
    mapper->addMapping(ui->label, 12);
    mapper->addMapping(ui->cbConnectionStatus, 13);
}

void AddMoney::addMoney()
{
    mapper->submit();
    ui->spBalance->setValue(0);
    emit applyChanges();
    close();
}

void AddMoney::updateTime()
{
    QString date = QDateTime::currentDateTime().toString("dd.MM.yyyy,
hh:mm:ss");
    ui->label->setText(date);
}

void AddMoney::compare()
{
    int index = ui->cbConnectionStatus->currentIndex();
    switch(index) {
    case 0:
        ui->spBalance->setReadOnly(false);
        mapper->addMapping(ui->label, 12);
        ui->spBalance->setMinimum(1);
        break;
    case 1:
        ui->spBalance->setReadOnly(false);
        ui->spBalance->setMinimum(0);
        break;
    case 2:
        ui->spBalance->setMinimum(-1);
        ui->spBalance->setValue(-1);
        ui->spBalance->setReadOnly(true);
        break;
    }
}

```

database.cpp

```

#include "database.h"

Database::Database(QObject *parent) : QObject(parent)
{
}

```

```

void Database::connectToDatabase()
{
    QString dbLocation = QApplication::applicationDirPath() +
"/customers.sqlite";
    db = QSqlDatabase::addDatabase("SQLITE");
    db.setDatabaseName(dbLocation);
    if(!db.open()) {
        qDebug() << "Помилка підключення до БД за замовчуванням: " <<
db.lastError().text();
        return;
    } else {
        qDebug() << "Успішне підключення до БД за замовчуванням";
    }
}

void Database::connectToDatabase(QString &filePath)
{
    db = QSqlDatabase::addDatabase("SQLITE");
    db.setDatabaseName(filePath);
    if(!db.open()) {
        qDebug() << "Помилка підключення до заданої БД: " <<
db.lastError().text();
        return;
    } else {
        qDebug() << "Успішне підключення до заданої БД";
    }
}

void Database::insertIntoTable(QVariantList &data)
{
    query.prepare("INSERT INTO Customers (SURNAME, NAME, PATRONYMIC, BALANCE,
CITY, STREET, BUILDING, APARTMENT, TYPE, TARIFF, CONNECTED, LAST_PAY, STATUS)
"
                "VALUES (:SURNAME, :NAME, :PATRONYMIC, :BALANCE, :CITY,
:SREET, :BUILDING, :APARTMENT, :TYPE, :TARIFF, :CONNECTED, :LAST_PAY,
:STATUS)");
    query.bindValue(":SURNAME", data[0].toString());
    query.bindValue(":NAME", data[1].toString());
    query.bindValue(":PATRONYMIC", data[2].toString());
    query.bindValue(":BALANCE", data[3].toInt());
    query.bindValue(":CITY", data[4].toString());
    query.bindValue(":STREET", data[5].toString());
    query.bindValue(":BUILDING", data[6].toInt());
    query.bindValue(":APARTMENT", data[7].toInt());
    query.bindValue(":TYPE", data[8].toString());
    query.bindValue(":TARIFF", data[9].toString());
    query.bindValue(":CONNECTED", data[10].toString());
    query.bindValue(":LAST_PAY", data[11].toString());
    query.bindValue(":STATUS", data[12].toString());
    if(!query.exec()) {
        qDebug() << "Помилка додавання даних в БД: " << query.lastError();
        return;
    }
}

QList<QString> Database::getPaymentsHistory()
{
    QList<QString> payments;
    if(!query.exec("SELECT * FROM Customers WHERE LAST_PAY")) {
        qDebug() << "Помилка отримання даних з БД: " << query.lastError();
    }
    while(query.next()) {
        QString surname = query.value("SURNAME").toString();
        QString name = query.value("NAME").toString();
        QString patronymic = query.value("PATRONYMIC").toString();
        QString pay = query.value("LAST_PAY").toString();
        QString record = pay + " -> " + surname + " " + name + " " +
patronymic;
        payments.append(record);
    }
    return payments;
}

```

```

QList<QString> Database::getDebtors ()
{
    QList<QString> debtors;
    if(!query.exec("SELECT * FROM Customers WHERE STATUS = 'Відключений за борги'")) {
        qDebug() << "Помилка отримання даних з БД: " << query.lastError();
    }
    while(query.next()) {
        QString surname = query.value("SURNAME").toString();
        QString name = query.value("NAME").toString();
        QString patronymic = query.value("PATRONYMIC").toString();
        QString record = surname + " " + name + " " + patronymic;
        debtors.append(record);
    }
    return debtors;
}

QList<QString> Database::getCities(QString city)
{
    QList<QString> cities;
    if(!query.exec(QString("SELECT * FROM Customers WHERE CITY = '%1'").arg(city))) {
        qDebug() << "Помилка отримання даних з БД: " << query.lastError();
    }
    while(query.next()) {
        QString surname = query.value("SURNAME").toString();
        QString name = query.value("NAME").toString();
        QString patronymic = query.value("PATRONYMIC").toString();
        QString record = surname + " " + name + " " + patronymic;
        cities.append(record);
    }
    qDebug() << query.lastQuery();
    return cities;
}

void Database::dropTable ()
{
    db.open();
    QSqlQuery q(db);
    if(!q.exec("DELETE FROM Customers")) {
        qDebug() << "Помилка видалення таблиці: " << query.lastError();
    }
}

```

editcustomer.cpp

```

#include "editcustomer.h"
#include "ui_editcustomer.h"

EditCustomer::EditCustomer(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::EditCustomer)
{
    ui->setupUi(this);
    mapper = new QDataWidgetMapper(this);
    mapper->setSubmitPolicy(QDataWidgetMapper::ManualSubmit);
    connect(ui->pbApply, SIGNAL(clicked(bool)), this, SLOT(check()));
    connect(ui->pbCancel, SIGNAL(clicked(bool)), this, SLOT(close()));
    QRegExp expression("[A-ЯІЄ{1}][a-яєії]{39}");
    QRegExpValidator *validator = new QRegExpValidator(expression, this);
    ui->lineName->setValidator(validator);
    ui->lineSurname->setValidator(validator);
    ui->lineStreet->setValidator(validator);
}

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

void EditCustomer::setModel(QAbstractItemModel *model)
{

```

```

        mapper->setModel(model);
        mapper->addMapping(ui->lineSurname, 1);
        mapper->addMapping(ui->lineName, 2);
        mapper->addMapping(ui->linePatronymic, 3);
        mapper->addMapping(ui->cbCities, 5);
        mapper->addMapping(ui->lineStreet, 6);
        mapper->addMapping(ui->spBuilding, 7);
        mapper->addMapping(ui->spApartment, 8);
        mapper->addMapping(ui->cbConnectionTypes, 9);
        mapper->addMapping(ui->cbTariffs, 10);
    }

    void EditCustomer::editCustomer()
    {
        mapper->submit();
        emit applyChanges();
        close();
    }

    void EditCustomer::check()
    {
        if(ui->lineName->text().length() < 2
            || ui->lineSurname->text().length() < 2
            || ui->linePatronymic->text().length() < 2
            || ui->lineStreet->text().length() < 2)
            QMessageBox::warning(this, "Помилка вводу", "Перевірте введені дані");
        else
            this->editCustomer();
    }

```

main.cpp

```

#include "mainwindow.h"
#include <QApplication>

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

    return a.exec();
}

```

mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    msgBox = new QMessageBox(this);
    db = new Database(this);
    this->openDatabase();
    model = new QSqlTableModel(this);
    edit = new EditCustomer(this);
    add = new AddCustomer(this);
    money = new AddMoney(this);
    output = new OutputDocuments(this);
    this->setupModel();
    this->setupWindow();
    this->setConnects();
}

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

```

```

QSqlTableModel *MainWindow::getModel()
{
    return model;
}

void MainWindow::search(const QString &searchValue)
{
    model->setFilter(QString("SURNAME LIKE '%%1%'").arg(searchValue));
    if(searchValue.isEmpty()) {
        model->setFilter("");
        this->selectRow();
    }
}

void MainWindow::addCustomer()
{
    add->exec();
    this->selectRow();
}

void MainWindow::removeCustomer()
{
    int selectedRow = ui->tableView->currentIndex().row();
    if(selectedRow >= 0) {
        msgBox->setWindowTitle("Видалення запису");
        msgBox->setIcon(QMessageBox::Question);
        msgBox->setText("Ви впевнені?");
        msgBox->setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
        int res = msgBox->exec();
        if(res == QMessageBox::Ok) {
            model->removeRow(selectedRow);
            model->select();
            resizeTable();
            emit customerRemoved();
        } else {
            return;
        }
    }
}

void MainWindow::editCustomer()
{
    edit->mapper->setCurrentModelIndex(ui->tableView->currentIndex());
    edit->exec();
}

void MainWindow::addMoney()
{
    money->mapper->setCurrentModelIndex(ui->tableView->currentIndex());
    money->exec();
}

void MainWindow::resizeTable()
{
    ui->tableView->resizeColumnsToContents();
}

void MainWindow::updateTable()
{
    model->select();
}

void MainWindow::clearTable()
{
    msgBox->setWindowTitle("Очищення таблиці");
    msgBox->setIcon(QMessageBox::Question);
    msgBox->setText("Ви впевнені?");
    msgBox->setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
    int res = msgBox->exec();
    if(res == QMessageBox::Ok) {
        db->dropTable();
        model->select();
    }
}

```



```

    } else {
        return;
    }
}

void MainWindow::contextMenu(QPoint pos)
{
    QMenu *menu = new QMenu(this);
    QAction *deleteCustomer = new QAction(tr("Видалити"), this);
    QAction *editCustomer = new QAction(tr("Редагувати"), this);
    QAction *addMoney = new QAction(tr("Керування балансом"), this);
    connect(deleteCustomer, SIGNAL(triggered(bool)), this,
    SLOT(removeCustomer()));
    connect(editCustomer, SIGNAL(triggered(bool)), this,
    SLOT(editCustomer()));
    connect(addMoney, SIGNAL(triggered(bool)), this, SLOT(addMoney()));
    menu->addAction(editCustomer);
    menu->addAction(addMoney);
    menu->addAction(deleteCustomer);
    menu->popup(ui->tableView->mapToGlobal(pos));
}

void MainWindow::selectRow()
{
    ui->tableView->selectRow(0);
}

void MainWindow::setupModel()
{
    model->setTable("Customers");
    model->select();
    edit->setModel(model);
    money->setModel(model);
    model->setHeaderData(ID, Qt::Horizontal, QObject::tr("ID"));
    model->setHeaderData(SURNAME, Qt::Horizontal, QObject::tr("Прізвище"));
    model->setHeaderData(NAME, Qt::Horizontal, QObject::tr("Ім'я"));
    model->setHeaderData(PATRONYMIC, Qt::Horizontal, QObject::tr("По
    батькові"));
    model->setHeaderData(BALANCE, Qt::Horizontal, QObject::tr("Баланс"));
    model->setHeaderData(CITY, Qt::Horizontal, QObject::tr("Місто"));
    model->setHeaderData(STREET, Qt::Horizontal, QObject::tr("Вулиця"));
    model->setHeaderData(BUILDING, Qt::Horizontal, QObject::tr("Будинок"));
    model->setHeaderData(APARTMENT, Qt::Horizontal, QObject::tr("Квартира"));
    model->setHeaderData(TYPE, Qt::Horizontal, QObject::tr("Тип
    підключення"));
    model->setHeaderData(TARIFF, Qt::Horizontal, QObject::tr("Тариф"));
    model->setHeaderData(CONNECTED, Qt::Horizontal,
    QObject::tr("Підключений"));
    model->setHeaderData(LAST_PAY, Qt::Horizontal, QObject::tr("Остання
    оплата"));
    model->setHeaderData(STATUS, Qt::Horizontal, QObject::tr("Статус"));
}

void MainWindow::openDatabase()
{
    QString dbLocation = QApplication::applicationDirPath() +
    "/customers.sqlite";
    if(QFile(dbLocation).exists()) {
        db->connectToDatabase();
        this->setWindowTitle("Відкрита база даних - " + dbLocation);
    } else {
        msgBox->setWindowTitle("Не вдалося знайти БД за замовчуванням");
        msgBox->setIcon(QMessageBox::Question);
        msgBox->setText("Не вдалося знайти БД за замовчуванням\nВказати шлях
    до БД вручну?");
        msgBox->setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
        int res = msgBox->exec();
        switch(res) {
            case QMessageBox::Ok: {
                QString path = QFileDialog::getOpenFileName(this, tr("Вибір БД
    вручну"), "", tr("База даних (*.sqlite)"));
                db->connectToDatabase(path);
                this->setWindowTitle("Відкрита база даних - " + path);
                break;
            }
        }
    }
}

```

```

        }
        case QMessageBox::Cancel:
            exit(0);
    }
}

void MainWindow::setupWindow()
{
    ui->mainToolBar->setMovable(false);
    ui->mainToolBar->setContextMenuPolicy(Qt::PreventContextMenu);
    ui->tableView->setModel(model);
    ui->tableView->setColumnHidden(0, true);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
    ui->tableView->setSelectionMode(QAbstractItemView::SingleSelection);
    ui->tableView->resizeColumnsToContents();
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->horizontalHeader()->setStretchLastSection(true);
    ui->tableView->setContextMenuPolicy(Qt::CustomContextMenu);
    ui->tableView->setSortingEnabled(true);
    ui->tableView->selectRow(0);
}

void MainWindow::setConnects()
{
    connect(ui->lineSearch, SIGNAL(textChanged(QString)), this,
    SLOT(search(QString)));
    connect(ui->tableView, SIGNAL(customContextMenuRequested(QPoint)), this,
    SLOT(contextMenu(QPoint)));
    connect(ui->mtbAddCustomer, SIGNAL(triggered(bool)), this,
    SLOT(addCustomer()));
    connect(ui->mtbEditCustomer, SIGNAL(triggered(bool)), this,
    SLOT(editCustomer()));
    connect(ui->mtbAddMoney, SIGNAL(triggered(bool)), this, SLOT(addMoney()));
    connect(ui->mtbRemoveCustomer, SIGNAL(triggered(bool)), this,
    SLOT(removeCustomer()));
    connect(ui->mtbDropTable, SIGNAL(triggered(bool)), this,
    SLOT(clearTable()));
    connect(ui->mtbOutputDocuments, SIGNAL(triggered(bool)), output,
    SLOT(exec()));
    connect(ui->tableView, SIGNAL(doubleClicked(QModelIndex)), this,
    SLOT(editCustomer()));
    connect(add, SIGNAL(applyChanges()), this, SLOT(updateTable()));
    connect(add, SIGNAL(applyChanges()), this, SLOT(resizeTable()));
    connect(add, SIGNAL(applyChanges()), this, SLOT(selectRow()));
    connect(this, SIGNAL(customerRemoved()), this, SLOT(selectRow()));
}

```

outputdocuments.cpp

```

#include "outputdocuments.h"
#include "ui_outputdocuments.h"

OutputDocuments::OutputDocuments(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::OutputDocuments)
{
    ui->setupUi(this);
    db = new Database(this);
    this->getDebtorsList();
    this->getPaymentsHistory();
    this->getSortedList(0);
    connect(ui->tabWidget, SIGNAL(currentChanged(int)), this,
    SLOT(check(int)));
    connect(ui->pbUpdate, SIGNAL(clicked(bool)), this, SLOT(update()));
    connect(ui->pbPaymentsHistory, SIGNAL(clicked(bool)), this,
    SLOT(savePaymentsHistory()));
    connect(ui->pbDebtors, SIGNAL(clicked(bool)), this,
    SLOT(saveDebtorsList()));
    connect(ui->pbPrice, SIGNAL(clicked(bool)), this, SLOT(savePriceList()));
    connect(ui->cbSities, SIGNAL(currentIndexChanged(int)), this,
    SLOT(getSortedList(int)));
}

```

```

        connect(ui->pbSort, SIGNAL(clicked(bool)), this, SLOT(saveSortedList()));
    }

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

void OutputDocuments::getPaymentsHistory()
{
    ui->listPaymentsHistory->clear();
    QList<QString> payments = db->getPaymentsHistory();
    QString item;
    foreach (item, payments) {
        ui->listPaymentsHistory->addItem(item);
    }
}

void OutputDocuments::getDebtorsList()
{
    ui->listDebtors->clear();
    QList<QString> debtors = db->getDebtors();
    QString item;
    foreach (item, debtors) {
        ui->listDebtors->addItem(item);
    }
}

void OutputDocuments::savePaymentsHistory()
{
    QString fileName = QFileDialog::getSaveFileName(this,
                                                    tr("Зберегти історію
платежів"), "",
                                                    tr("Log File (*.log)"));

    if(fileName.isEmpty()) {
        return;
    } else {
        QFile file(fileName);
        file.open(QFile::WriteOnly | QFile::Text);
        QTextStream stream(&file);
        QList<QString> payments = db->getPaymentsHistory();
        QString item;
        QString buffer;
        foreach (item, payments) {
            buffer += item + '\n';
        }
        stream << buffer;
    }
}

void OutputDocuments::saveDebtorsList()
{
    QString fileName = QFileDialog::getSaveFileName(this,
                                                    tr("Зберегти список
боржників"), "",
                                                    tr("Log File (*.log)"));

    if(fileName.isEmpty()) {
        return;
    } else {
        QFile file(fileName);
        file.open(QFile::WriteOnly | QFile::Text);
        QTextStream stream(&file);
        QList<QString> debtors = db->getDebtors();
        QString item;
        QString buffer;
        foreach (item, debtors) {
            buffer += item + '\n';
        }
        stream << buffer;
    }
}

void OutputDocuments::getSortedList(int city)
{

```



```

    if(fileName.isEmpty()) {
        return;
    } else {
        QFile list(fileName);
        list.open(QFile::WriteOnly | QFile::Text);
        QTextStream stream(&list);
        int cities = ui->spCities->value();
        int privateSector = ui->spPrivateSector->value();
        int masterCost = ui->spMaster->value();
        int ethernet = ui->spEthernet->value();
        int os = ui->spOperatingSystem->value();
        int hardware = ui->spHardware->value();
        QString price = "Вартість підключення (багатоповерхівки): " +
QString::number(cities) + "\n" +
        "Вартість підключення (приватний сектор): " +
QString::number(privateSector) + "\n" +
        "Виклик фахівця на об'єкт: " + QString::number(masterCost) +
"\n" +
        "Метр звитої пари: " + QString::number(ethernet) + "\n" +
        "Налаштування ОС: " + QString::number(os) + "\n" +
        "Налаштування обладнання: " + QString::number(hardware);
        stream << price;
        list.close();
    }
}

void OutputDocuments::check(int index)
{
    if(index > 1)
        ui->pbUpdate->setEnabled(false);
    else
        ui->pbUpdate->setEnabled(true);
}

void OutputDocuments::update()
{
    this->getPaymentsHistory();
    this->getDebtorsList();
}

```

ДОДАТОК В – ЛІСТИНГ РЕАЛІЗАЦІЇ ІНТЕРФЕЙСУ

addcustomer.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>AddCustomer</class>
  <widget class="QDialog" name="AddCustomer">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>524</width>
        <height>305</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Додавання запису</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="0" column="1">
        <widget class="QLineEdit" name="lineSurname"/>
      </item>
      <item row="0" column="0">
        <widget class="QLabel" name="label">
          <property name="text">
            <string>Прізвище:</string>
          </property>
        </widget>
      </item>
      <item row="1" column="0">
        <widget class="QLabel" name="label_2">
          <property name="text">
            <string>Ім'я:</string>
          </property>
        </widget>
      </item>
      <item row="2" column="0">
        <widget class="QLabel" name="label_3">
          <property name="text">
            <string>По батькові:</string>
          </property>
        </widget>
      </item>
      <item row="1" column="1">
        <widget class="QLineEdit" name="lineName">
          <property name="text">
            <string/>
          </property>
        </widget>
      </item>
      <item row="3" column="0">
        <widget class="QLabel" name="label_4">
          <property name="text">
            <string>Баланс:</string>
          </property>
        </widget>
      </item>
      <item row="2" column="1">
        <widget class="QLineEdit" name="linePatronymic"/>
      </item>
      <item row="4" column="0">
        <widget class="QLabel" name="label_5">
          <property name="text">
            <string>Місто:</string>
          </property>
        </widget>
      </item>
      <item row="5" column="0">

```

```

<widget class="QLabel" name="label_6">
  <property name="text">
    <string>Вулиця:</string>
  </property>
</widget>
</item>
<item row="3" column="1">
  <layout class="QHBoxLayout" name="horizontalLayout_2">
    <item>
      <widget class="QCheckBox" name="cbMoney">
        <property name="text">
          <string>Додати гроші на рахунок</string>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QSpinBox" name="spBalance">
        <property name="maximum">
          <number>1200</number>
        </property>
      </widget>
    </item>
  </layout>
</item>
<item row="6" column="0">
  <widget class="QLabel" name="label_7">
    <property name="text">
      <string>Будинок:</string>
    </property>
  </widget>
</item>
<item row="5" column="1">
  <widget class="QLineEdit" name="lineStreet"/>
</item>
<item row="7" column="0">
  <widget class="QLabel" name="label_8">
    <property name="text">
      <string>Квартира:</string>
    </property>
  </widget>
</item>
<item row="7" column="1">
  <widget class="QSpinBox" name="spApartment">
    <property name="minimum">
      <number>1</number>
    </property>
    <property name="maximum">
      <number>1000</number>
    </property>
  </widget>
</item>
<item row="6" column="1">
  <widget class="QSpinBox" name="spBuilding">
    <property name="minimum">
      <number>1</number>
    </property>
    <property name="maximum">
      <number>1000</number>
    </property>
  </widget>
</item>
<item row="8" column="0">
  <widget class="QLabel" name="label_9">
    <property name="text">
      <string>Тип:</string>
    </property>
  </widget>
</item>
<item row="8" column="1">
  <widget class="QComboBox" name="cbConnectionTypes">
    <item>

```

```

    <property name="text">
      <string>Звита пара</string>
    </property>
  </item>
  <item>
    <property name="text">
      <string>Оптоволокно</string>
    </property>
  </item>
</widget>
</item>
<item row="9" column="1">
  <widget class="QComboBox" name="cbTariffs">
    <item>
      <property name="text">
        <string>40 Мбіт/с</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>80 Мбіт/с</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>100 Мбіт/с</string>
      </property>
    </item>
  </widget>
</item>
<item row="9" column="0">
  <widget class="QLabel" name="label_10">
    <property name="text">
      <string>Тариф:</string>
    </property>
  </widget>
</item>
<item row="10" column="1">
  <layout class="QHBoxLayout" name="horizontalLayout">
    <item>
      <spacer name="horizontalSpacer">
        <property name="orientation">
          <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
          <size>
            <width>40</width>
            <height>20</height>
          </size>
        </property>
      </spacer>
    </item>
    <item>
      <widget class="QPushButton" name="pbApply">
        <property name="text">
          <string>Додати</string>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QPushButton" name="pbCancel">
        <property name="text">
          <string>Відміна</string>
        </property>
      </widget>
    </item>
  </layout>
</item>
<item row="4" column="1">
  <widget class="QComboBox" name="cbCities">
    <item>

```



```

        <property name="text">
            <string>Мариуполь</string>
        </property>
    </item>
    <item>
        <property name="text">
            <string>Покровськ</string>
        </property>
    </item>
    <item>
        <property name="text">
            <string>Авдіївка</string>
        </property>
    </item>
</widget>
</item>
</layout>
</widget>
<tabstops>
    <tabstop>lineSurname</tabstop>
    <tabstop>lineName</tabstop>
    <tabstop>linePatronymic</tabstop>
    <tabstop>cbMoney</tabstop>
    <tabstop>spBalance</tabstop>
    <tabstop>cbCities</tabstop>
    <tabstop>lineStreet</tabstop>
    <tabstop>spBuilding</tabstop>
    <tabstop>spApartment</tabstop>
    <tabstop>cbConnectionTypes</tabstop>
    <tabstop>cbTariffs</tabstop>
    <tabstop>pbApply</tabstop>
    <tabstop>pbCancel</tabstop>
</tabstops>
<resources/>
<connections/>
</ui>

```

addmoney.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>AddMoney</class>
    <widget class="QDialog" name="AddMoney">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>196</width>
                <height>127</height>
            </rect>
        </property>
        <property name="windowTitle">
            <string>Оплата</string>
        </property>
        <layout class="QGridLayout" name="gridLayout_2">
            <item row="0" column="0">
                <layout class="QHBoxLayout" name="horizontalLayout_2">
                    <item>
                        <spacer name="horizontalSpacer_2">
                            <property name="orientation">
                                <enum>Qt::Horizontal</enum>
                            </property>
                            <property name="sizeHint" stdset="0">
                                <size>
                                    <width>40</width>
                                    <height>20</height>
                                </size>
                            </property>
                        </spacer>
                    </item>
                    <item>

```

```

<widget class="QLabel" name="label">
  <property name="text">
    <string>TextLabel</string>
  </property>
</widget>
</item>
<item>
  <spacer name="horizontalSpacer_3">
    <property name="orientation">
      <enum>Qt::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
      <size>
        <width>40</width>
        <height>20</height>
      </size>
    </property>
  </spacer>
</item>
</layout>
</item>
<item row="1" column="0">
  <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="0">
      <layout class="QVBoxLayout" name="verticalLayout_2">
        <item>
          <widget class="QLabel" name="label_3">
            <property name="text">
              <string>Статус:</string>
            </property>
          </widget>
        </item>
        <item>
          <widget class="QLabel" name="label_2">
            <property name="text">
              <string>Баланс:</string>
            </property>
          </widget>
        </item>
      </layout>
    </item>
    <item row="0" column="1">
      <layout class="QVBoxLayout" name="verticalLayout">
        <item>
          <widget class="QComboBox" name="cbConnectionStatus">
            <item>
              <property name="text">
                <string>Підключений</string>
              </property>
            </item>
            <item>
              <property name="text">
                <string>Відключений</string>
              </property>
            </item>
            <item>
              <property name="text">
                <string>Відключений за борги</string>
              </property>
            </item>
          </widget>
        </item>
        <item>
          <widget class="QSpinBox" name="spBalance">
            <property name="minimum">
              <number>-1</number>
            </property>
            <property name="maximum">
              <number>1200</number>
            </property>
          </widget>
        </item>
      </layout>
    </item>
  </layout>
</item>

```

```

        </item>
    </layout>
</item>
</layout>
</item>
<item row="2" column="0">
    <layout class="QHBoxLayout" name="horizontalLayout">
        <item>
            <spacer name="horizontalSpacer">
                <property name="orientation">
                    <enum>Qt::Horizontal</enum>
                </property>
                <property name="sizeHint" stdset="0">
                    <size>
                        <width>40</width>
                        <height>20</height>
                    </size>
                </property>
            </spacer>
        </item>
        <item>
            <widget class="QPushButton" name="pbApply">
                <property name="text">
                    <string>OK</string>
                </property>
            </widget>
        </item>
        <item>
            <widget class="QPushButton" name="pbCancel">
                <property name="text">
                    <string>Відміна</string>
                </property>
            </widget>
        </item>
    </layout>
</item>
</layout>
</widget>
<resources/>
<connections/>
</ui>

```

editcustomer.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>EditCustomer</class>
    <widget class="QDialog" name="EditCustomer">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>496</width>
                <height>277</height>
            </rect>
        </property>
        <property name="windowTitle">
            <string>Редагування запису</string>
        </property>
        <layout class="QGridLayout" name="gridLayout">
            <item row="1" column="0">
                <widget class="QLabel" name="label_2">
                    <property name="text">
                        <string>Ім'я:</string>
                    </property>
                </widget>
            </item>
            <item row="9" column="1">
                <layout class="QHBoxLayout" name="horizontalLayout">
                    <item>

```

```

<spacer name="horizontalSpacer">
  <property name="orientation">
    <enum>Qt::Horizontal</enum>
  </property>
  <property name="sizeHint" stdset="0">
    <size>
      <width>599</width>
      <height>20</height>
    </size>
  </property>
</spacer>
</item>
<item>
  <widget class="QPushButton" name="pbApply">
    <property name="text">
      <string>OK</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QPushButton" name="pbCancel">
    <property name="text">
      <string>Відміна</string>
    </property>
  </widget>
</item>
</layout>
</item>
<item row="2" column="0">
  <widget class="QLabel" name="label_3">
    <property name="text">
      <string>По батькові:</string>
    </property>
  </widget>
</item>
<item row="4" column="0">
  <widget class="QLabel" name="label_6">
    <property name="text">
      <string>Вулиця:</string>
    </property>
  </widget>
</item>
<item row="0" column="1">
  <widget class="QLineEdit" name="lineSurname"/>
</item>
<item row="2" column="1">
  <widget class="QLineEdit" name="linePatronymic"/>
</item>
<item row="3" column="0">
  <widget class="QLabel" name="label_5">
    <property name="text">
      <string>Місто:</string>
    </property>
  </widget>
</item>
<item row="1" column="1">
  <widget class="QLineEdit" name="lineName">
    <property name="text">
      <string/>
    </property>
  </widget>
</item>
<item row="0" column="0">
  <widget class="QLabel" name="label">
    <property name="text">
      <string>Прізвище:</string>
    </property>
  </widget>
</item>
<item row="8" column="0">
  <widget class="QLabel" name="label_10">

```

```

    <property name="text">
      <string>Тариф:</string>
    </property>
  </widget>
</item>
<item row="6" column="1">
  <widget class="QSpinBox" name="spApartment">
    <property name="minimum">
      <number>1</number>
    </property>
    <property name="maximum">
      <number>1000</number>
    </property>
  </widget>
</item>
<item row="5" column="0">
  <widget class="QLabel" name="label_7">
    <property name="text">
      <string>Будинок:</string>
    </property>
  </widget>
</item>
<item row="8" column="1">
  <widget class="QComboBox" name="cbTariffs">
    <item>
      <property name="text">
        <string>40 Мбіт/с</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>80 Мбіт/с</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>100 Мбіт/с</string>
      </property>
    </item>
  </widget>
</item>
<item row="7" column="0">
  <widget class="QLabel" name="label_9">
    <property name="text">
      <string>Тип:</string>
    </property>
  </widget>
</item>
<item row="7" column="1">
  <widget class="QComboBox" name="cbConnectionTypes">
    <item>
      <property name="text">
        <string>Звита пара</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Оптоволокно</string>
      </property>
    </item>
  </widget>
</item>
<item row="4" column="1">
  <widget class="QLineEdit" name="lineStreet"/>
</item>
<item row="5" column="1">
  <widget class="QSpinBox" name="spBuilding">
    <property name="minimum">
      <number>1</number>
    </property>
    <property name="maximum">

```

```

        <number>1000</number>
    </property>
</widget>
</item>
<item row="6" column="0">
    <widget class="QLabel" name="label_8">
        <property name="text">
            <string>Квартира:</string>
        </property>
    </widget>
</item>
<item row="3" column="1">
    <widget class="QComboBox" name="cbCities">
        <item>
            <property name="text">
                <string>Маріуполь</string>
            </property>
        </item>
        <item>
            <property name="text">
                <string>Покровськ</string>
            </property>
        </item>
        <item>
            <property name="text">
                <string>Авдіївка</string>
            </property>
        </item>
    </widget>
</item>
</layout>
</widget>
<resources/>
<connections/>
</ui>

```

mainwindow.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>MainWindow</class>
    <widget class="QMainWindow" name="MainWindow">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>1388</width>
                <height>667</height>
            </rect>
        </property>
        <property name="windowTitle">
            <string>Hello World</string>
        </property>
        <widget class="QWidget" name="centralWidget">
            <layout class="QGridLayout" name="gridLayout_2">
                <item row="0" column="0">
                    <widget class="QLabel" name="labelSearch">
                        <property name="text">
                            <string>Пошук за прізвищем</string>
                        </property>
                    </widget>
                </item>
                <item row="0" column="1">
                    <widget class="QLineEdit" name="lineSearch"/>
                </item>
                <item row="1" column="0" colspan="2">
                    <widget class="QGroupBox" name="groupBox">
                        <property name="title">
                            <string>База даних клієнтів</string>
                        </property>

```

```

        <layout class="QGridLayout" name="gridLayout">
            <item row="0" column="0">
                <widget class="QTableView" name="tableView"/>
            </item>
        </layout>
    </widget>
</item>
</layout>
</widget>
<widget class="QMenuBar" name="menuBar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>1388</width>
            <height>21</height>
        </rect>
    </property>
</widget>
<widget class="QToolBar" name="mainToolBar">
    <attribute name="toolBarArea">
        <enum>TopToolBarArea</enum>
    </attribute>
    <attribute name="toolBarBreak">
        <bool>>false</bool>
    </attribute>
    <addaction name="mtbAddCustomer"/>
    <addaction name="mtbEditCustomer"/>
    <addaction name="mtbAddMoney"/>
    <addaction name="mtbRemoveCustomer"/>
    <addaction name="separator"/>
    <addaction name="mtbOutputDocuments"/>
    <addaction name="separator"/>
    <addaction name="mtbDropTable"/>
</widget>
<widget class="QStatusBar" name="statusBar"/>
<action name="mtbAddCustomer">
    <property name="text">
        <string>Додати</string>
    </property>
</action>
<action name="mtbEditCustomer">
    <property name="text">
        <string>Редагувати</string>
    </property>
</action>
<action name="mtbRemoveCustomer">
    <property name="text">
        <string>Видалити</string>
    </property>
</action>
<action name="mtbAddMoney">
    <property name="text">
        <string>Керування балансом</string>
    </property>
</action>
<action name="mtbOutputDocuments">
    <property name="text">
        <string>Вихідні документи</string>
    </property>
</action>
<action name="mtbDropTable">
    <property name="text">
        <string>Очистити таблицю</string>
    </property>
</action>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>

```

outputdocuments.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>OutputDocuments</class>
  <widget class="QDialog" name="OutputDocuments">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>585</width>
        <height>347</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Вихідні документи</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="0" column="0">
        <widget class="QTabWidget" name="tabWidget">
          <property name="currentIndex">
            <number>0</number>
          </property>
          <widget class="QWidget" name="tab">
            <attribute name="title">
              <string>Історія платежів</string>
            </attribute>
            <layout class="QGridLayout" name="gridLayout_2">
              <item row="0" column="0">
                <widget class="QListWidget" name="listPaymentsHistory"/>
              </item>
              <item row="1" column="0">
                <widget class="QPushButton" name="pbPaymentsHistory">
                  <property name="text">
                    <string>Зберегти в файл</string>
                  </property>
                </widget>
              </item>
            </layout>
          </widget>
          <widget class="QWidget" name="tab_2">
            <attribute name="title">
              <string>Список боржників</string>
            </attribute>
            <layout class="QGridLayout" name="gridLayout_3">
              <item row="0" column="0">
                <widget class="QListWidget" name="listDebtors"/>
              </item>
              <item row="1" column="0">
                <widget class="QPushButton" name="pbDebtors">
                  <property name="text">
                    <string>Зберегти в файл</string>
                  </property>
                </widget>
              </item>
            </layout>
          </widget>
          <widget class="QWidget" name="tab_3">
            <attribute name="title">
              <string>Сортування</string>
            </attribute>
            <layout class="QGridLayout" name="gridLayout_8">
              <item row="0" column="0">
                <widget class="QLabel" name="label_7">
                  <property name="text">
                    <string>Вивести всіх клієнтів з міста:</string>
                  </property>
                </widget>
              </item>
            </layout>
          </widget>
        </widget>
      </item>
    </layout>
  </widget>
</ui>

```



```

</item>
<item row="0" column="1" colspan="2">
  <widget class="QComboBox" name="cbSities">
    <item>
      <property name="text">
        <string>Мариуполь</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Покровськ</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Авдіївка</string>
      </property>
    </item>
  </widget>
</item>
<item row="1" column="0" colspan="3">
  <widget class="QListWidget" name="listSort"/>
</item>
<item row="2" column="0" colspan="3">
  <widget class="QPushButton" name="pbSort">
    <property name="text">
      <string>Зберегти в файл</string>
    </property>
  </widget>
</item>
</layout>
</widget>
<widget class="QWidget" name="tab_4">
  <attribute name="title">
    <string>Прайс-лист</string>
  </attribute>
  <layout class="QGridLayout" name="gridLayout_6">
    <item row="0" column="0">
      <widget class="QGroupBox" name="groupBox">
        <property name="title">
          <string/>
        </property>
        <layout class="QGridLayout" name="gridLayout_7">
          <item row="0" column="0">
            <widget class="QLabel" name="label">
              <property name="text">
                <string>Вартість підключення (багатоповерхівки):</string>
              </property>
            </widget>
          </item>
          <item row="0" column="1">
            <widget class="QSpinBox" name="spCities">
              <property name="minimum">
                <number>1</number>
              </property>
              <property name="maximum">
                <number>2000</number>
              </property>
            </widget>
          </item>
          <item row="1" column="0">
            <widget class="QLabel" name="label_2">
              <property name="text">
                <string>Вартість підключення (приватний сектор):</string>
              </property>
            </widget>
          </item>
          <item row="1" column="1">
            <widget class="QSpinBox" name="spPrivateSector">
              <property name="minimum">
                <number>1</number>
              </property>
            </widget>
          </item>
        </layout>
      </groupBox>
    </item>
  </layout>
</widget>

```

```

        </property>
        <property name="maximum">
            <number>2000</number>
        </property>
    </widget>
</item>
<item row="2" column="0">
    <widget class="QLabel" name="label_3">
        <property name="text">
            <string>Виклик фахівця на об'єкт:</string>
        </property>
    </widget>
</item>
<item row="2" column="1">
    <widget class="QSpinBox" name="spMaster">
        <property name="minimum">
            <number>1</number>
        </property>
        <property name="maximum">
            <number>2000</number>
        </property>
    </widget>
</item>
<item row="3" column="0">
    <widget class="QLabel" name="label_4">
        <property name="text">
            <string>Метр звитої пари:</string>
        </property>
    </widget>
</item>
<item row="3" column="1">
    <widget class="QSpinBox" name="spEthernet">
        <property name="minimum">
            <number>1</number>
        </property>
        <property name="maximum">
            <number>2000</number>
        </property>
    </widget>
</item>
<item row="4" column="0">
    <widget class="QLabel" name="label_5">
        <property name="text">
            <string>Налаштування ОС:</string>
        </property>
    </widget>
</item>
<item row="4" column="1">
    <widget class="QSpinBox" name="spOperatingSystem">
        <property name="minimum">
            <number>1</number>
        </property>
        <property name="maximum">
            <number>2000</number>
        </property>
    </widget>
</item>
<item row="5" column="0">
    <widget class="QLabel" name="label_6">
        <property name="text">
            <string>Налаштування обладнання:</string>
        </property>
    </widget>
</item>
<item row="5" column="1">
    <widget class="QSpinBox" name="spHardware">
        <property name="minimum">
            <number>1</number>
        </property>
        <property name="maximum">
            <number>2000</number>
        </property>
    </widget>
</item>

```

```

        </property>
      </widget>
    </item>
  </layout>
</widget>
</item>
<item row="1" column="0">
  <widget class="QPushButton" name="pbPrice">
    <property name="text">
      <string>Зберегти в файл</string>
    </property>
  </widget>
</item>
</layout>
</widget>
</widget>
</item>
<item row="1" column="0">
  <widget class="QPushButton" name="pbUpdate">
    <property name="text">
      <string>Оновити</string>
    </property>
  </widget>
</item>
</layout>
</widget>
<resources/>
<connections/>
</ui>

```