Міністерство освіти і науки України

Одеський національний політехнічний університет

Херсонський політехнічний фаховий коледж

програмний додаток

«Облік послуг на підприємстві з ремонту та

пошиття одягу»

Пояснювальна записка до курсового проекту

ХПФК.000352.01.П2.М -ЛУ

Аркушів …

**В курсовому проекті використано ідеї, результати і тексти інших авторів**

**та є посилання на відповідні джерела**

Керівник проекту

Т.Є. Багмет

.05.2022

Розробив студент

Д.О.Білий

.05.2022

Нормоконтролер

С.О. Бабикін

.05.2022

2022

ЗМІСТ

С.

[ВСТУП 5](#_Toc103534457)

[2 ПОСТАНОВКА ЗАДАЧІ 6](#_Toc103534458)

[2.1 Характеристика предметної області 6](#_Toc103534459)

[2.2 Вимоги до програми 6](#_Toc103534460)

[2.3 Структура вхідних даних 8](#_Toc103534461)

[2.4 Структура вихідних даних 9](#_Toc103534462)

[3 КОМ’ЮТЕРНА СИСТЕМА 10](#_Toc103534463)

[3.1 Технічні характеристики комп’ютера та зовнішніх пристроїв 10](#_Toc103534464)

[3.2 Вибір програмних засобів та операційної системи 11](#_Toc103534465)

[4 ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ 13](#_Toc103534466)

[4.1 Інтерфейс програми 13](#_Toc103534467)

[4.2 Алгоритм рішення задачі 14](#_Toc103534468)

[5 ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ 16](#_Toc103534469)

[5.1 Розробка програми 16](#_Toc103534470)

[5.2 Етапи відладки 17](#_Toc103534471)

[5.3 Типи помилок 17](#_Toc103534472)

[ВИСНОВКИ 19](#_Toc103534473)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 20](#_Toc103534474)

[Додаток А Схема алгоритму програми 21](#_Toc103534475)

[Додаток Б Лістинг програми 22](#_Toc103534476)

[Додаток В Роздрук результатів роботи програми 49](#_Toc103534477)

# ВСТУП

На протязі всієї своєї історії люди завжди потребували і потребують одягу власного особливого стилю. Люди завжди хотіли відрізнятись від інших. Кожен зокрема хотів змайструвати власну річ, унікальну, але не мали потрібних для цього навичок та вмінь пошиття. Проте існують майстри своєї праці, які виконують цю непросту роботу – створюють новітні моделі одягу. Але таким задачам потрібна чудова організація та ефективний облік усіх цих замовлень.

Тому конкретно програмний додаток «Облік замовлень ательє з пошиття та ремонту одягу» створений для обліку замовлень, де співробітник може сформувати заказ, вказавши параметри клієнта, з якого матеріалу буде зроблений, та який колір буде мати.

Робота підприємства ательє з пошиву та ремонту одягу є доволі актуальною в наш час, адже сьогодні важко уявити себе без модного одягу, новинок, унікальних вбрань та комфортних костюмів. Люди, які слідкують за собою, завжди хочуть бути першими і мати власний стиль в одязі. Тому розробка саме такої системи обліку та адміністрування замовлень в майстернях з пошиву та ремонту одягу суттєво поліпшує роботу, адже допомагає вчасно, а головне  
якісно вести облік замовлень, заказаних клієнтами.

З огляду на те, програмний додаток розробляється з метою автоматизування монотонної роботи співробітника із базою даних, що має зняти зі співробітника велику кількість рутинних справ зв’язаних із обробкою та аналізом даних, забезпечити швидкий доступ до створення звітів та виводу їх на друк.

Дане програмне забезпечення можна застосовувати в будь-якій компанії, яка займається наданням послуг ремонту та пошиття одягу. Глави компанії завжди повинні відслідковувати роботу компанії і стежити за своїми співробітниками. Для цього потрібна спільна база даних, що включає всю необхідну інформацію та програма, яка буде зрозуміла і автоматизує роботу з базою даних.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Характеристика предметної області

Ательє — майстерня індивідуального пошиву одягу, аксесуарів різних напрямків та стилів. Інформаційна підсистема «Облік замовлень ательє з пошиття та ремонту одягу» призначена саме для обліку замовлень, де співробітник може без зайвих проблем створити замовлення, вказавши надані клієнтом параметри (матеріал виробу, розмір, колір). Також співробітник має можливість робити огляд замовлення клієнтів, видаляти або за потреби редагувати їх, а також формувати звітність по наданим послугам.

Головними напрямками діяльності ательє з пошиву та ремонту одягу є: індивідуальне пошиття чоловічого та жіночого одягу, ремонт одягу будь-якої складності, підгонка виробу по фігурі клієнта.

Користувач також має змогу робити пошук по всім наявним в базі даних таблицях використовуючи при цьому коди об’єктів (клієнтів, працівників, послуг, тощо).

Для забезпечення цілісності даних потрібно реалізувати якісну обробку помилок, які можуть виникнути під час роботи користувача з програмним додатком. В іншому випадку користувач може отримати неочікуваний результат виконання тої, чи іншої частини коду. Тому, після виконання кожної операції з базою даних буде реалізований обробник подій, який забезпечить стабільність роботи програми.

2.2 Вимоги до програми

Програмне забезпечення призначене лише для адміністрації ательє, тому доступ до даних повинен бути захищений. Для цього буде використовуватися логін та пароль, встановлений в програмі.

До головних функцій програми буде входити:

1. Можливість реєстрації клієнта, про якого є такі дані:

* код клієнта;
* ПІБ;
* телефон;

1. Реєстрація працівників, що містить:

* код працівника;
* ПІБ працівника;
* стаж;
* стать працівника;
* телефон;
* зарплату

1. Реєстрація продуктів, що містить:

* код продукту;
* тип продукту;
* назву;
* код матеріалу;
* колір;
* фото продукту;
* вартість;

1. Реєстрація послуг ремонту, що містить:

* код послуги;
* назву;
* вартість;

1. Реєстрація матеріалів, що містить:

* код матеріалу;
* назву;
* виробника;

1. Реєстрація заказів, що містить:

* код заказу;
* код продукту;
* код клієнта;
* код працівника
* дату початку
* дату закінчення
* ціну
* статус оплати

1. Можливість пошуку клієнта по його коду;
2. Пошук по номеру заказу;
3. Видалення клієнта, працівника, матеріалу, заказу чи послуги;
4. Пошук працівника по його коду;
5. Пошук послуг та матеріалів за їх кодами;

Програмне забезпечення розробляється для операційних систем Windows (7/8/10) та Linux на мові програмування Python з використання бази даних Sqlite3. ПЗ повинно забезпечувати легке оновлення інформації, відображення якої передбачається цим технічним завданням.

2.3 Структура вхідних даних

Структура вхідних даних представлена у таблиці 2.1.

Таблиця 2.1 Структура вхідних даних

|  |  |  |  |
| --- | --- | --- | --- |
| Ідентифікатор | Тип | Діапазон значень | Пояснення |
| app | Pyqt5 Application |  | Основне вікно |
| widget | Pyqt5 Qwidget |  | Стартовий віджет |
| conn | Sqlite3.connection |  | Зв’язок з БД |
| Query | String |  | Запит до БД |

Продовження таблиці 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| name\_client | String |  | ПІБ Клієнта |
| phone\_client | String |  | Телефон клієнта |
| name\_worker | String |  | ПІБ працівника |
| phone\_worker | String |  | Телефон працівника |
| stash\_worker | String |  | Стаж працівника |
| gender\_worker | String | м, ж | Стать працівника |
| salary\_worker | String | 0….999999 | Зарплата працівника |
| name\_material | String |  | Назва матеріалу |
| producer\_material | String |  | Виробник матеріалу |
| name\_product | String |  | Назва виробу |
| code\_material | Int |  | Код матеріалу |
| color\_product | String |  | Колір виробу |
| photo\_product | Bytes |  | Фото виробу |
| price\_product | Int | 0…99999 | Ціна виробу |
| type\_product | String |  | Тип виробу |
| name\_repair | String |  | Назва послуги ремонту |
| price\_repair | Int | 0…99999 | Вартість послуги ремонту |
| payed\_offer | Int | 0, -1 | Статус оплати заказу |
| code\_client | Int |  | Код клієнта |
| code\_worker | Int |  | Код працівника |
| price\_offer | Int | 0…99999 | Націнка заказу |
| date\_of\_start | String |  | Дата початку заказу |
| date\_of\_end | String |  | Дата закінчення заказу |
| product\_code | Int |  | Код продукту |

2.4 Структура вихідних даних

Структура вихідних даних представлена у таблиці 2.2.

Таблиця 2.2 Структура вихідних даних

|  |  |  |  |
| --- | --- | --- | --- |
| Ідентифікатор | Тип | Діапазон значень | Пояснення |
| code\_client | Int |  | Код клієнта |
| name\_client | String |  | ПІБ клієнта |
| phone\_client | String |  | Телефон клієнта |

Продовження таблиці 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| code\_offer | Int |  | Код заказу |
| code\_worker | Int |  | Код працівника |
| code\_product | Int |  | Код продукту |
| date\_of\_start | String |  | Дата початку заказу |
| date\_of\_end | String |  | Дата закінчення заказу |
| price\_offer | Int | 0…999999 | Націнка заказу |
| payed\_offer | Int | 0, -1 | Статус оплати заказу |
| name\_worker | String |  | ПІБ працівника |
| stash\_worker | Int | 0...100 | Стаж працівника |
| gender\_worker | String | м,ж | Стать працівника |
| salary\_worker | Int | 0…999999 | Зарплата працівника |
| phone\_worker | String |  | Телефон працівника |
| code\_material | Int |  | Код матеріалу |
| name\_material | String |  | Назва матеріалу |
| producer\_material | String |  | Назва виробника |
| name\_product | String |  | Назва виробу |
| price\_product | Int | 0…999999 | Ціна виробу |
| type\_product | String |  | Тип виробу |
| color\_product | String |  | Колір виробу |
| photo\_product | Bytes |  | Фото виробу |
| price\_repair | Int | 0…999999 | Ціна послуги ремонту |
| name\_repair | String |  | Назва послуги ремонту |

3 КОМП’ЮТЕРНА СИСТЕМА

3.1 Технічні характеристики комп’ютера та зовнішніх пристроїв

Курсовий проект розроблявся на персональному комп’ютері з наступними технічною характеристиками:

* Процесор AMD Ryzen 3400G;
* Оперативна пам’ять 16 ГБ DDR4 3200 MHz;
* Відеокарта AMD Radeon Vega 11;
* Монітор Xiaomi 1920x1080 60Hz;
* Операційна система Windows 10 Pro;

Дана програма запускається та справно працювати на комп’ютерах з наступними технічними характеристиками:

* Процесор з частотою 2.0 GHz;
* Об’єм оперативної пам’яті 1024МБ;
* Відеокарта об’ємом 512 МБ;
* Операційна система починаючи з Windows 7;

3.2 Вибір програмних засобів та операційної системи

Програмне забезпечення по своєму призначенню підрозділяють на системне, прикладне та інструментальне.

Системне програмне забезпечення – це комплекс програм для управління ресурсами ПК (ЦП, пам'яттю, вводом і виводом даних), підтримання працездатності системи обробки інформації, підвищення ефективності її використання.

Цей клас програмного забезпечення носить загальний характер застосування і не залежить від предметної області. До системного програмного забезпечення пред'являються високі вимоги надійності роботи, зручності та ефективності використання. Як правило, комп'ютер купується з встановленим системним програмним забезпеченням. Системне програмне забезпечення включає в себе базову та сервісне програмне забезпечення.

Операційна система - великий набір програм для управління взаємодією всіх компонентів комп'ютера в процесі роботи і виконання поставлених завдань. Забезпечує роботу і взаємодію комп'ютерних пристроїв в цілому як системи, координує зв'язок з іншими пристроями в мережі, а користувачам і прикладним програмам надає ресурси і інтерфейс - засіб взаємодії з комп'ютером.

Середовищем розробки програмного забезпечення для розробки курсового проєкту було обрано Visual Studio Code, бо середовище містить великий функціонал та можливості для розробки, дозволяючи працювати з любою мовою програмування за пару кліків. Visual Studio Code має сучасний зручний інтерфейс з чуйним дизайном, який можна налаштувати по своєму бажанню. Також можна зазначити що це середовище має низькі технічні вимоги, що дозволяє йому запускатися і комфортно працювати на бюджетних комп’ютерах. Для розробки цього програмного забезпечення було вирішено обрати мову програму Python. Я вважаю, що ця мова має багато плюсів порівняно з іншими

Переваги роботи з мовою Python:

* висока швидкість розробки ПЗ порівняно з мовами більш низького рівня;
* мова є дуже гнучкою, якщо чогось недостатньо в стандартній комплектації, завжди можна це дописати власноруч;
* читати код написаний на Python дуже легко, зменшується кількість «спагетті-коду»;
* простота синтаксису, базові операції можна вивчити менш ніж за день;
* велика спільнота розробників, та власні правила («дзен-пітону») написання коду на цій мові.

Для розробки курсового проекту було вирішено обрати операційну систему Windows, тому що це найбільш розповсюджена і комфортна операційна система. Операційна система має досить великий функціонал і зручна у роботі. Більшість користувачів ПК обирають саме операційну систему Windows. Переваги операційної системи Windows:

* зручність та привабливість інтерфейсу;
* для її використання не потрібно додаткових знань в області комп’ютерної науки.
* система досить проста у використанні;
* більшість ПЗ як платного, так і безкоштовного підтримують саме цю операційну систему.
* якщо виникають питання, найбільш ймовірно знайти відповіді в мережі, знову ж таки через її поширеність;
* ця операційна система дуже проста в налаштуванні, і її можна використовувати вже після встановлення.

4 ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ

4.1 Інтерфейс програми

Цінність будь-якої програми визначається не тільки її здатністю безпомилково і якісно вирішувати поставлену задачу, а й тим, наскільки легко і зручно цією програмою можна користуватися. Навіть найпростіша, і в той же час, функціональна програма виявиться незатребуваною, якщо користувач (тобто кінцевий споживач програми) при роботі з нею швидко втомлюється, дратується, відчуває себе некомфортно. У цьому випадку користувач зазвичай робить більше помилок і працює менш продуктивно (тобто неефективно). Для отримання дійсно зручної програми, з якою б було приємно працювати, необхідний добре продуманий дружній інтерфейс з користувачем. Під призначеним для користувача інтерфейсом (англ. User interface) будемо розуміти своєрідний комунікаційний канал (сполучна ланка), за допомогою якого користувач взаємодіє з комп'ютером.

Інтерфейс є найважливішою частиною будь-якої програми, так як саме з інтерфейсом користувач стикається в процесі своєї роботи. З точки зору користувача саме інтерфейс є кінцевим продуктом. Інтерфейс являє собою сукупність використовуваних в програмі засобів для введення даних, способів відображення інформації на екрані дисплея, елементів для управління різними етапами рішення задачі. Основне призначення (головна мета) призначеного для користувача інтерфейсу - забезпечити короткий і зручний шлях до отримання результату.

Невдалі інтерфейси сильно уповільнюють роботу користувачів, сприяють швидкої стомлюваності, великому числу помилок і, як наслідок, відбивають у користувачів подальше бажання працювати з програмою. У зв'язку з цим розробнику програми слід враховувати відомі рекомендації провідних фахівців в області створення призначеного для користувача інтерфейсу.

Саме тому було розроблено максимально функціональний і зручний для користування інтерфейс. Для розробки інтерфейсу були використані стандартні для оточення Windows кольори та відтінки(див. рис. 4.1)

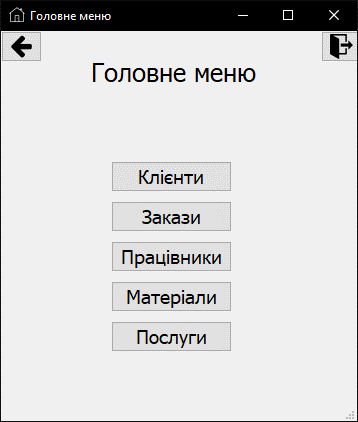


Рисунок 4.1 – Головне меню

4.2 Алгоритм рішення задачі

Програма в даному курсовому проекті складається з декількох екранних форм, що забезпечують зручну роботу. Кожна форма реалізована за допомогою класу, що складається зі своїх власних методів та подій.

Спершу відкривається стартова форма, де описується головна інформація про виконавця та міститься єдина кнопка для переходу на головну форму та подальшої роботи з програмою.

self.submit\_button.clicked.connect(self.add\_func) – виконує запит на додавання нового запису до бази даних (клієнта, замовлення, працівника, послуги, тощо);

self.submit\_button.clicked.connect(self.edit\_func) – виконує запит на оновлення запису в базі даних;

self.delete\_button.clicked.connect(self.delete\_func) – виконує запит на видалення запису з бази даних;

self.find\_button.clicked.connect(self.search\_func) – виконує функцію пошуку в базі даних;

self.add\_new\_worker\_button.clicked.connect(self.to\_add\_worker\_func) – завантажує форму для додавання нового працівника;

self.view\_workers\_button.clicked.connect(self.to\_workers\_list\_func) – завантажує форму для перегляду списку робітників;

self.workers\_analisys\_button.clicked.connect(self.to\_workers\_report\_func) – завантажує форму для перегляду звітності по працівникам;

self.view\_clients\_button.clicked.connect(self.to\_clients\_list\_func) – завантажує форму для перегляду списку клієнтів;

self.view\_products\_button.clicked.connect(self.to\_products\_list\_func) – завантажує форму для перегляду списку виробів;

self.view\_offers\_button.clicked.connect(self.to\_offers\_list\_func) – завантажує форму для перегляду списку замовлень;

self.add\_new\_client\_button.clicked.connect(self.to\_add\_client\_func) – завантажує форму для додавання нового клієнта;

self.add\_new\_offer\_button.clicked.connect(self.to\_add\_offer\_func) – завантажує форму для створення замовлення;

self.add\_new\_product\_button.clicked.connect(self.to\_add\_product\_func) – завантажує форму для додавання нового виробу;

5 ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ

5.1 Розробка програми

Програма, що була розроблена в ході роботи над курсовим проєктом може працювати як на операційних системах Microsoft Windows так і на ОС з ядром Linux (для цього потрібно провести компіляцію програми на Linux).

Після запуску програми, користувачу відкриється стартова форма з інформацією про виконавця курсового проєкту. Для переходу на головну форму потрібно натиснути відповідну кнопку на стартовій формі.На головній формі в меню доступні наступні пункти:

* Клієнтами;
* Працівники;
* Послуги;
* Закази;
* Матеріали.

Обравши пункт “Клієнти”, користувач переходить на форму для роботи з клієнтами. На даній формі користувачу надається змога додавати клієнта, переглядати їх список, отримати звітність або ж видалити клієнта з бази даних.

Для повернення в головне меню користувачу потрібно натиснути кнопку зі стрілкою «Назад».

Обравши пункт “Працівники”, користувач переходить на форму для роботи з робітниками ательє. Користувач має змогу додати, видалити або переглянути наявних у базі робітників ательє. Також є можливість переглянути аналіз роботи працівників та провести друк цього документу.

Обравши пункт “Послуги”, користувач переходить на форму для роботи з послугами, що надає ательє. Користувач має змогу переглянути наявні послуги ремонту та продукти, видаляти їх або ж за бажаннями додавати нові

Обравши пункт “Закази”, користувач переходить на форму для роботи з замовленнями. Для користувача відкривається можливість працювати з замовленнями ательє, додавати, видаляти, відтворювати пошук та видавати чек.

5.2 Етапи відладки

Відкладка - це процес виявлення причин виникнення помилки та її подальшого виправлення (на відміну від тестування, що є процесом виявлення самого факту існування помилки).

Відкладка включає в себе елементи тестування і розробки. На деяких проектах відкладка може займати до 50% всього часу розробки. Для багатьох програмістів відкладка - це найважча частина програмування. Процес відкладки включає в себе:

* Вироблення системних вимог;
* Вироблення вимог до програмного забезпечення;
* Аналіз та проектування;
* Реалізацію - процес написання коду;
* Продуктизацію – процес вияснення відношення програмного забезпечення вимогам і доведення до них;
* Експлуатацію – процес використання програми.

5.3 Типи помилок

Існують різні типи програмних помилок, які можуть виникати на етапі розробки програми програмного забезпечення. Можна привести деякі з них:

* синтаксичні помилки - легко виявляється компіляторами і інтерпретаторами мови і легко виправляється;
* помилки виконання - в програмі виникає виняток, це означає, що трапилося непередбачене;
* семантичні помилки - це застосування операторів, які не дають потрібного ефекту;

У курсовому проекті зустрічалися помилки кожного з типів, які були успішно виправлені.

ВИСНОВКИ

Для успішного виконання курсового проєкту була використана мова програмування Python, через її легкість в освоєнні та швидкість програмування.

Візуальна частина курсового проєкту була виконана з використання ПО для візуального проектування QT Designer.

Програмне забезпечення може бути використане ательє, які хочуть мінімізувати рутину під час роботи з базою даних та підвищення продуктивності працівників під час цього процесу.

Під час розробки вдалося реалізувати всі заплановані функції, що спростило її використання компаніями. Завдяки інтуїтивно зрозумілого інтерфейсу користувач може почати користуватися програмою без попереднього ознайомлення з документацією.

Під час роботи над курсовим проєктом було вивчене застосування складних SQL запитів, інструментів візуального програмування в середі QT Designer, а також фреймворку PyQt5 для розробки користувацьких інтерфейсів на мові Python.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Марк Лутц. Изучаем Python (5 издание) 1-2 томы.
2. [Python 3 и PyQt 5. Разработка приложений (2016)](http://www.ozon.ru/context/detail/id/136151723/)
3. И.В.Горячая. Пользовательский интерфейс: назначение, принципы разработки и практические рекомендации.

https://al.cs.msu.ru/files/goryachaya.user.interface.2010.pdf

1. Типы ошибок в программном обеспечении.

https://studfile.net/preview/2672406/page:38/

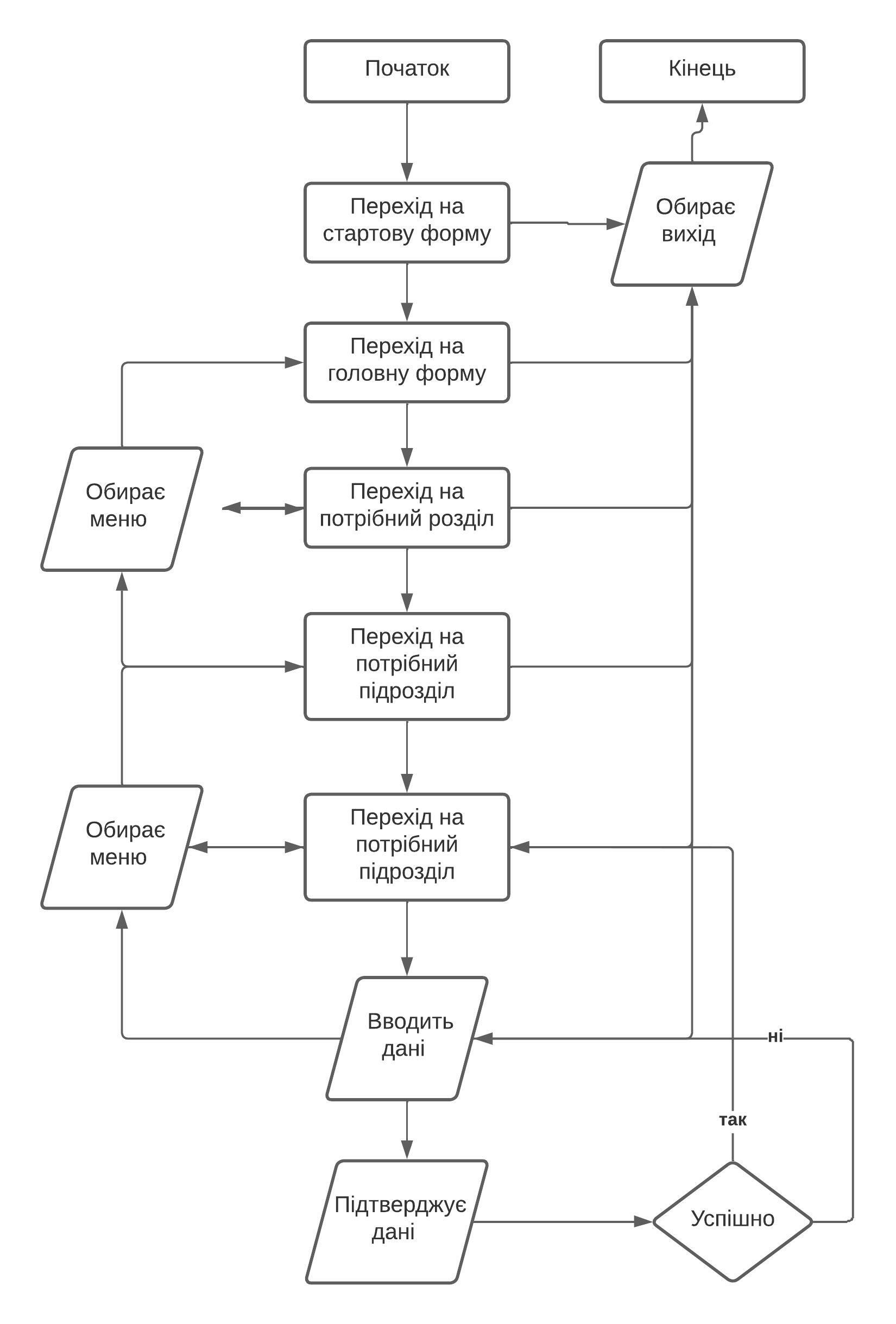
1. Основные принципы отладки программного обеспечения.

https://intellect.icu/vidy-oshibok-i-osnovnye-printsipy-otladki-programmnogo-obespecheniya-6184

1. Багмет Т.Є. Методичні вказівки щодо виконання курсового проекту з дисципліни “Об’єктно орієнтоване програмування”.

Додаток А

Схема алгоритму програми



Додаток Б

Лістинг програми

# -\*- coding: utf-8 -\*-

from reportlab.lib.enums import TA\_CENTER

from reportlab.platypus import Paragraph, SimpleDocTemplate, Table, TableStyle

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.pdfbase import pdfmetrics

from reportlab.pdfbase.ttfonts import TTFont

from reportlab.lib.styles import ParagraphStyle

from reportlab.platypus import Paragraph

from reportlab.lib.pagesizes import A4

from reportlab.lib import colors

from PyQt5.QtCore import Qt

from PySide2.QtCore import \*

from PySide2.QtGui import \*

from PySide2.QtWidgets import \*

import sys

import os

import resource\_rc # импорт ресурсов --- создается с помощью ---- pyrcc5 resources.qrc -o resource\_rc.py

import sqlite3

from PySide2.QtCore import Qt

import re

# Глобальный выход

def exit\_func():

print('exiting...')

sys.exit()

# Стартовое меню

class start(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.to\_menu\_button.clicked.connect(self.to\_main\_menu\_func)

self.setWindowIcon(QIcon(u":/newPrefix/assets/start\_menu\_icon.png"))

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def setupUi(self):

self.setWindowTitle('Стартове меню')

uic.loadUi("uis/start.ui", self)

# retranslateUi

# Главное меню

class main\_menu(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/start\_menu\_icon.png"))

self.back\_button.clicked.connect(self.to\_start\_func)

self.exit\_button.clicked.connect(exit\_func)

self.to\_clients\_button.clicked.connect(self.to\_clients\_func)

self.to\_workers\_button.clicked.connect(self.to\_workers\_func)

self.to\_services\_button.clicked.connect(self.to\_services\_func)

self.to\_materials\_button.clicked.connect(self.to\_materials\_func)

self.to\_offers\_button.clicked.connect(self.to\_offers\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def to\_materials\_func(self):

self.cams = materials()

self.cams.show()

self.close()

def to\_services\_func(self):

self.cams = services()

self.cams.show()

self.close()

def to\_workers\_func(self):

self.cams = workers()

self.cams.show()

self.close()

def to\_clients\_func(self):

self.cams = clients()

self.cams.show()

self.close()

def to\_start\_func(self):

self.cams = start()

self.cams.show()

self.close()

def setupUi(self):

self.setWindowTitle('Головне меню')

uic.loadUi("uis/main\_menu.ui", self)

# Меню клиентов

class clients(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/clients\_icon.png"))

self.back\_button.clicked.connect(self.to\_main\_menu\_func)

self.exit\_button.clicked.connect(exit\_func)

self.add\_new\_client\_button.clicked.connect(self.to\_add\_client\_func)

self.view\_clients\_button.clicked.connect(self.to\_clients\_list\_func)

self.clients\_report\_button.clicked.connect(self.to\_clients\_report\_func)

def to\_clients\_report\_func(self):

self.cams = clients\_report()

self.cams.show()

self.close()

def to\_clients\_list\_func(self):

self.cams = clients\_list()

self.cams.show()

self.close()

def to\_add\_client\_func(self):

self.cams = add\_client()

self.cams.show()

self.close()

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def setupUi(self):

self.setWindowTitle('Меню Клієнти')

uic.loadUi("uis/clients", self)

# Добавить клиента

class add\_client(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.back\_button.clicked.connect(self.to\_clients\_func)

self.exit\_button.clicked.connect(exit\_func)

self.submit\_button.clicked.connect(self.add\_client\_func)

self.phone\_field.setInputMask('999-999')

def add\_client\_func(self):

conn = sqlite3.connect("atelie.db")

name = self.name\_field.text()

phone = self.phone\_field.text()

if name == '' or phone == '' or phone == '-' or len(phone) < 7:

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

row = (name, phone)

# print('DATA =', row) # for debugging

query = '''insert into Клієнти (ПІБ, Телефон)

values (?, ?);'''

conn.execute(query, row)

self.name\_field.clear()

self.phone\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

self.name\_field.clear()

self.phone\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def to\_clients\_func(self):

self.cams = clients()

self.cams.show()

self.close()

def setupUi(self):

self.setWindowTitle('Додати клієнта')

uic.loadUi("window.ui", self)

# Редактировать клиента

class edit\_client(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.submit\_button.clicked.connect(self.edit\_func)

self.cancel\_button.clicked.connect(self.cancel\_func)

self.code = None

def set\_code(self, code):

self.code = code

def cancel\_func(self):

self.close()

def edit\_func(self):

name = self.name\_field.text()

phone = self.phone\_field.text()

code = self.code

print('DATA = ',name,phone,code)

if name == '' or phone == '-' or len(phone) < 7:

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

conn = sqlite3.connect("atelie.db")

with conn:

query = '''UPDATE Клієнти SET ПІБ = ?, Телефон = ? WHERE Код\_клієнта = ?;'''

conn.execute(query, (name,phone,code) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.close()

def setupUi(self):

uic.loadUi("uis/edit\_clien.ui", self)

self.setWindowTitle('Редагувати запис')

# Просмотр клиенто

class clients\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_clients\_func)

self.exit\_button.clicked.connect(exit\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.update\_button.clicked.connect(self.load\_data\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.edit\_button.clicked.connect(self.edit\_func)

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_client()

self.cams.set\_code(code=code)

self.cams.show()

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def delete\_func(self):

# current\_row = self.tableWidget.currentRow()

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Клієнти WHERE Код\_клієнта=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.load\_data\_func()

def to\_clients\_func(self):

self.cams = clients()

self.cams.show()

self.close()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT \* FROM Клієнти"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(row[1]))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(row[2]))

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

# print('data loaded ') for debugging

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

# print('err') for debugging

def setupUi(self):

uic.loadUi("uis/clients\_list.ui", self)

self.setWindowTitle('Список клієнтів')

self.tableWidget.setColumnCount(3)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(1,319)

self.tableWidget.setColumnWidth(2,270)

self.tableWidget.setHorizontalHeaderLabels(["Код клієнта","ПІБ","Телефон"])

self.tableWidget.verticalHeader().setVisible(False)

self.load\_data\_func()

# Меню работников

class workers(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/worker\_icon.png"))

self.back\_button.clicked.connect(self.to\_main\_menu\_func)

self.exit\_button.clicked.connect(exit\_func)

self.view\_workers\_button.clicked.connect(self.to\_workers\_list\_func)

self.add\_new\_worker\_button.clicked.connect(self.to\_add\_worker\_func)

self.workers\_analisys\_button.clicked.connect(self.to\_workers\_report\_func)

def to\_workers\_report\_func(self):

self.cams = workers\_report()

self.cams.show()

self.close()

def to\_add\_worker\_func(self):

self.cams = add\_workers()

self.cams.show()

self.close()

def to\_workers\_list\_func(self):

self.cams = workers\_list()

self.cams.show()

self.close()

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/workers.ui", self)

self.setWindowTitle("Робітники")

# Просмотр работников

class workers\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_workers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.update\_button.clicked.connect(self.load\_data\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.edit\_button.clicked.connect(self.edit\_func)

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_worker()

self.cams.set\_code(code)

self.cams.show()

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

# print('err') for debugging

def delete\_func(self):

current\_row = self.tableWidget.currentRow()

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Працівники WHERE Код\_працівника=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT \* FROM Працівники"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])+' ₴'))

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

# print('data loaded ') for debugging

def to\_workers\_func(self):

self.cams = workers()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/workers\_list.ui", self)

self.setWindowTitle("Список робітників")

if (self.tableWidget.columnCount() < 3):

self.tableWidget.setColumnCount(3)

self.tableWidget.setObjectName(u"tableWidget")

self.tableWidget.setGeometry(QRect(80, 50, 671, 291))

self.tableWidget.setColumnCount(6)

self.tableWidget.setColumnWidth(1,250)

self.tableWidget.setColumnWidth(2,50)

self.tableWidget.setColumnWidth(4,80)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setHorizontalHeaderLabels(["Код працівника","ПІБ","Стаж","Телефон","Стать","Зарплата"])

self.tableWidget.verticalHeader().setVisible(False)

self.load\_data\_func()

# Добавить работника

class add\_workers(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.back\_button.clicked.connect(self.to\_workers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.submit\_button.clicked.connect(self.add\_worker\_func)

self.phone\_field.setInputMask('999-999')

self.phone\_field.setMaxLength(7)

def to\_workers\_func(self):

self.cams = workers()

self.cams.show()

self.close()

def add\_worker\_func(self):

conn = sqlite3.connect("atelie.db")

name = self.name\_field.text()

stazh = self.stazh\_field.text()

sex = self.sex\_field.text()

salary = self.salary\_field.text()

phone = self.phone\_field.text()

try:

with conn:

row = (name, stazh, phone, sex, salary)

print('DATA =', row) # for debugging

query = '''insert into Працівники (ПІБ\_працівника, Стаж, Телефон, Стать, Зарплата)

values (?, ?, ?, ?, ?);'''

conn.execute(query, row)

self.name\_field.clear()

self.phone\_field.clear()

self.stazh\_field.clear()

self.sex\_field.clear()

self.salary\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

self.name\_field.clear()

self.phone\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def setupUi(self):

uic.loadUi("uis/add\_workers.ui", self)

self.setWindowTitle("Додати працівника")

# Редактировать работника

class edit\_worker(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.cancel\_button.clicked.connect(self.cancel\_func)

self.submit\_button.clicked.connect(self.edit\_worker\_func)

self.code = None

def set\_code(self, code):

self.code = code

def cancel\_func(self):

self.close()

def edit\_worker\_func(self):

name = self.name\_field.text()

phone = self.phone\_field.text()

stazh = self.stazh\_field.text()

salary = self.salary\_field.text()

code = self.code

if self.sex\_male\_rb.isChecked():

sex = 'м'

elif self.sex\_female\_rb.isChecked():

sex = 'ж'

else:

sex = ''

#print('DATA = ',name,stazh,phone,sex,salary,code) # for debugging

if name == '' or phone == '-' or len(phone) < 7 or stazh == '' or salary == '' or sex == '':

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

conn = sqlite3.connect("atelie.db")

with conn:

query = '''UPDATE Працівники SET ПІБ\_працівника = ?, Стаж = ?, Телефон = ?, Стать = ?, Зарплата = ? WHERE Код\_працівника = ?;'''

conn.execute(query, (name,stazh,phone,sex,salary,code) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.close()

def setupUi(self):

uic.loadUi("uis/edit\_worker.ui", self)

self.setWindowTitle("Редагування запису")

# Меню услуг

class services(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/service\_icon.png"))

self.back\_button.clicked.connect(self.to\_main\_menu\_func)

self.exit\_button.clicked.connect(exit\_func)

self.to\_view\_products\_button.clicked.connect(self.to\_products\_func)

self.to\_view\_repair\_services\_button.clicked.connect(self.to\_repairs\_func)

self.add\_product\_button.clicked.connect(self.to\_add\_products\_func)

self.add\_repair\_service\_button.clicked.connect(self.to\_add\_repairs\_func)

def to\_add\_repairs\_func(self):

self.cams = add\_repair()

self.cams.show()

self.close()

def to\_repairs\_func(self):

self.cams = repairs\_list()

self.cams.show()

self.close()

def to\_products\_func(self):

self.cams = products\_list()

self.cams.show()

self.close()

def to\_add\_products\_func(self):

self.cams = add\_product()

self.cams.show()

self.close()

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/services.ui", self)

self.setWindowTitle("Послуги")

# Меню материалов

class materials(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/material\_icon.png"))

self.back\_button.clicked.connect(self.to\_main\_menu\_func)

self.exit\_button.clicked.connect(exit\_func)

self.view\_materials\_button.clicked.connect(self.to\_view\_materials\_func)

self.add\_new\_material\_button.clicked.connect(self.to\_add\_material\_func)

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def to\_view\_materials\_func(self):

self.cams = materials\_list()

self.cams.show()

self.close()

def to\_add\_material\_func(self):

self.cams = add\_material()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/materials.ui", self)

self.setWindowTitle("Матеріали")

# Добавить материал

class add\_material(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.back\_button.clicked.connect(self.to\_materials\_func)

self.exit\_button.clicked.connect(exit\_func)

self.submit\_button.clicked.connect(self.add\_material\_func)

def to\_materials\_func(self):

self.cams = materials()

self.cams.show()

self.close()

def add\_material\_func(self):

conn = sqlite3.connect("atelie.db")

name = self.name\_field.text()

producer = self.producer\_field.text()

try:

with conn:

row = (name, producer)

print('DATA =', row) # for debugging

query = '''insert into Матеріали (Назва, Виробник)

values (?, ?);'''

conn.execute(query, row)

self.name\_field.clear()

self.producer\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

self.name\_field.clear()

self.producer\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def setupUi(self):

uic.loadUi("uis/add\_material.ui", self)

self.setWindowTitle("Додати матеріал")

# Список материалов

class materials\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_materials\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.edit\_button.clicked.connect(self.edit\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.update\_button.clicked.connect(self.update\_func)

self.tableWidget.clicked.connect(self.currention\_func)

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_material()

self.cams.set\_code(code)

self.cams.show()

def update\_func(self):

self.load\_data\_func()

def to\_materials\_func(self):

self.cams = materials()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def delete\_func(self):

# current\_row = self.tableWidget.currentRow()

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Матеріали WHERE Код\_матеріалу=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.load\_data\_func()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT \* FROM Матеріали"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(row[1]))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(row[2]))

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

# print('data loaded ') for debugging

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

# print('err') for debugging

def setupUi(self):

uic.loadUi("uis/materials\_list.ui", self)

self.setWindowTitle("Список матеріалів")

if (self.tableWidget.columnCount() < 3):

self.tableWidget.setColumnCount(3)

self.tableWidget.setColumnCount(3)

self.tableWidget.setColumnWidth(0,90)

self.tableWidget.setColumnWidth(1,300)

self.tableWidget.setColumnWidth(2,270)

self.tableWidget.setHorizontalHeaderLabels(["Код матеріалу","Назва","Виробник"])

self.tableWidget.verticalHeader().setVisible(False)

self.load\_data\_func()

# Редактировать материал

class edit\_material(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.cancel\_button.clicked.connect(self.to\_materials\_list\_func)

self.submit\_button.clicked.connect(self.edit\_func)

self.code = None

def set\_code(self, code):

self.code = code

def edit\_func(self):

name = self.name\_field.text()

producer = self.producer\_field.text()

code = self.code

#print('DATA = ',name,producer,code) # for debugging

if name == '' or producer == '':

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

conn = sqlite3.connect("atelie.db")

with conn:

query = '''UPDATE Матеріали SET Назва = ?, Виробник = ? WHERE Код\_матеріалу = ?;'''

conn.execute(query, (name,producer,code) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.close()

def to\_materials\_list\_func(self):

self.close()

def setupUi(self):

uic.loadUi("uis/edit\_material.ui", self)

self.setWindowTitle("Редагування запису")

# Добавить продукт

class add\_product(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.file\_dialog\_button.clicked.connect(self.file\_dialog\_func)

self.submit\_button.clicked.connect(self.add\_func)

self.exit\_button.clicked.connect(exit\_func)

self.data = self.back\_button.clicked.connect(self.to\_services\_func)

def to\_services\_func(self):

self.cams = services()

self.cams.show()

self.close()

def file\_dialog\_func(self):

fname = QFileDialog.getOpenFileName(self, 'Open file', '',"JPEG (\*.JPEG \*.jpeg \*.JPG \*.jpg \*.JPE \*.jpe \*JFIF \*.jfif);; PNG (\*.PNG \*.png);; GIF (\*.GIF \*.gif);; Bitmap Files (\*.BMP \*.bmp \*.DIB \*.dib);; TIFF (\*.TIF \*.tif \*.TIFF \*.tiff);; ICO (\*.ICO \*.ico)")[0]

f = open(fname, 'rb')

with f:

self.data = f.read()

image\_profile = QImage(fname) #QImage object

image\_profile = image\_profile.scaled(191,191, aspectRatioMode=Qt.KeepAspectRatio, transformMode=Qt.SmoothTransformation) # To scale image for example and keep its Aspect Ration

self.photo\_view.setPixmap(QPixmap.fromImage(image\_profile))

def add\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

list\_item = self.material\_list.currentText()

cur = conn.cursor()

row = cur.execute('SELECT Код\_матеріалу FROM Матеріали WHERE Назва =?',(str(list\_item),))

material = row.fetchall()[0][0]

except sqlite3.Error as e:

print(e)

print(e.args)

name = self.name\_field.text()

type\_of\_prod = self.type\_field.text()

type = 'Пошив\_одягу'

color = self.color\_field.text()

price = self.price\_field.text()

if name == '' or type\_of\_prod == '' or color == '' or price == '' or self.data == True:

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

row = (type,type\_of\_prod,name,material,color,self.data,price)

# print('DATA =', row) # for debugging

query = '''insert into Послуги (Тип\_послуги, Тип\_продукту, Назва, Код\_матеріалу, Колір, Фото\_продукту, Вартість)

values (?, ?, ?, ?, ?, ?, ?);'''

conn.execute(query, row)

self.name\_field.clear()

self.type\_field.clear()

self.color\_field.clear()

self.price\_field.clear()

self.photo\_view.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def load\_data\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

query = '''SELECT Назва FROM Матеріали;'''

cur = conn.cursor()

tablerow = 0

for row in cur.execute(query):

self.material\_list.insertItem(tablerow,row[0])

tablerow += 1

conn.close()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

conn.close()

def setupUi(self):

uic.loadUi("uis/add\_product.ui", self)

self.setWindowTitle("Додати продукт")

self.load\_data\_func()

# Список продуктов

class products\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_services\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.update\_button.clicked.connect(self.load\_data\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.edit\_button.clicked.connect(self.edit\_func)

self.tableWidget.doubleClicked.connect(self.show\_image\_func)

def show\_image\_func(self):

self.code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.popup = photo\_viewing(code=self.code)

self.popup.show()

def to\_services\_func(self):

self.cams = services()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def delete\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Послуги WHERE Код\_продукту=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

# print(e) #for debugging

# print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.load\_data\_func()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT Код\_продукту, Тип\_продукту, Назва, Код\_матеріалу, Колір, Фото\_продукту, Вартість FROM Послуги WHERE Код\_матеріалу IS NOT NULL;"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

image\_profile = QImage(u":/newPrefix/assets/screpka.png") #QImage object

image\_profile = image\_profile.scaled(15,20, aspectRatioMode=Qt.KeepAspectRatio, transformMode=Qt.SmoothTransformation) # To scale image for example and keep its Aspect Ration

self.pic\_label = QLabel()

self.pic\_label.setPixmap(QPixmap.fromImage(image\_profile))

self.pic\_label.setToolTip("Натисніть два рази <b>ЛКМ</b> для перегляду фото продукту")

self.tableWidget.setCellWidget(tablerow, 5, self.pic\_label)

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])+' ₴'))

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

# print('data loaded ') for debugging

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_product(code)

self.cams.show()

def setupUi(self):

uic.loadUi("uis/products\_list.ui", self)

self.setWindowTitle("Список продуктів")

if (self.tableWidget.columnCount() < 7):

self.tableWidget.setColumnCount(7)

self.tableWidget.setColumnCount(7)

self.tableWidget.setColumnWidth(4,70)

self.tableWidget.setColumnWidth(0,85)

self.tableWidget.setColumnWidth(1,150)

self.tableWidget.setColumnWidth(2,150)

self.tableWidget.setColumnWidth(5,40)

self.tableWidget.setHorizontalHeaderLabels(["Код продукту","Тип продукту","Назва","Код матеріалу","Колір",'Фото','Вартість'])

self.tableWidget.verticalHeader().setVisible(False)

self.load\_data\_func()

# Просмотр фото

class photo\_viewing(QMainWindow):

def \_\_init\_\_(self,code):

super().\_\_init\_\_()

self.setWindowIcon(QIcon(u":/newPrefix/assets/photo\_icon.png"))

self.code = code

self.setupUi(self,self.code)

self.back\_button.clicked.connect(self.to\_products\_list\_func)

def to\_products\_list\_func(self):

self.close()

def load\_photo\_func(self,code):

try:

conn = sqlite3.connect("atelie.db")

with conn:

query = '''SELECT Фото\_продукту FROM Послуги WHERE Код\_продукту=?;'''

cur = conn.cursor()

row = cur.execute(query, (code,))

row = row.fetchall()[0][0]

image\_profile = QPixmap() #QImage object

image\_profile.loadFromData(row)

image\_profile = image\_profile.scaled(371,371, aspectRatioMode=Qt.KeepAspectRatio, transformMode=Qt.SmoothTransformation) # To scale image for example and keep its Aspect Ration

self.picture.setPixmap(image\_profile)

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

def setupUi(self, code):

uic.loadUi("uis/view\_product\_photo.ui", self)

self.setWindowTitle("Перегляд фото")

self.load\_photo\_func(code)

# Редактирование продукта

class edit\_product(QMainWindow):

def \_\_init\_\_(self,code):

super().\_\_init\_\_()

self.code = code

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.file\_dialog\_button.clicked.connect(self.file\_dialog\_func)

self.back\_button.clicked.connect(self.to\_products\_list\_func)

self.submit\_button.clicked.connect(self.edit\_func)

def file\_dialog\_func(self):

fname = QFileDialog.getOpenFileName(self, 'Open file', '',"JPEG (\*.JPEG \*.jpeg \*.JPG \*.jpg \*.JPE \*.jpe \*JFIF \*.jfif);; PNG (\*.PNG \*.png);; GIF (\*.GIF \*.gif);; Bitmap Files (\*.BMP \*.bmp \*.DIB \*.dib);; TIFF (\*.TIF \*.tif \*.TIFF \*.tiff);; ICO (\*.ICO \*.ico)")[0]

f = open(fname, 'rb')

with f:

self.data = f.read()

image\_profile = QImage(fname) #QImage object

image\_profile = image\_profile.scaled(191,191, aspectRatioMode=Qt.KeepAspectRatio, transformMode=Qt.SmoothTransformation) # To scale image for example and keep its Aspect Ration

self.photo\_view.setPixmap(QPixmap.fromImage(image\_profile))

def edit\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

list\_item = self.material\_list.currentText()

cur = conn.cursor()

row = cur.execute('SELECT Код\_матеріалу FROM Матеріали WHERE Назва =?',(str(list\_item),))

material = row.fetchall()[0][0]

except sqlite3.Error as e:

print(e)

print(e.args)

name = self.name\_field.text()

type\_of\_prod = self.type\_field.text()

type = 'Пошив одягу'

color = self.color\_field.text()

price = self.price\_field.text()

if name == '' or type\_of\_prod == '' or color == '' or price == '' or self.data == True:

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

row = (type,type\_of\_prod,name,material,color,self.data,price,self.code)

# print('DATA =', row) # for debugging

query = '''UPDATE Послуги SET Тип\_послуги=?, Тип\_продукту=?, Назва=?, Код\_матеріалу=?, Колір=?, Фото\_продукту=?, Вартість=? WHERE Код\_продукту=?;'''

conn.execute(query, row)

self.name\_field.clear()

self.type\_field.clear()

self.color\_field.clear()

self.price\_field.clear()

self.photo\_view.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def load\_data\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

query = '''SELECT Назва FROM Матеріали;'''

cur = conn.cursor()

tablerow = 0

for row in cur.execute(query):

self.material\_list.insertItem(tablerow,row[0])

tablerow += 1

conn.close()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

conn.close()

def to\_products\_list\_func(self):

self.close()

def setupUi(self):

uic.loadUi("uis/edit\_product.ui", self)

self.setWindowTitle("Редагування запису")

self.load\_data\_func()

# Добавление услуги ремонта

class add\_repair(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.back\_button.clicked.connect(self.to\_services\_func)

self.exit\_button.clicked.connect(exit\_func)

self.submit\_button.clicked.connect(self.add\_func)

def add\_func(self):

conn = sqlite3.connect("atelie.db")

name = self.name\_field.text()

price = self.price\_field.text()

if name == '' or price == '':

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

row = ('Ремонт\_одягу',name, price)

# print('DATA =', row) # for debugging

query = '''insert into Послуги (Тип\_послуги,Тип\_продукту, Назва,Код\_матеріалу,Колір, Фото\_продукту, Вартість)

values (?, NULL, ?, NULL, NULL, NULL,?);'''

conn.execute(query, row)

self.name\_field.clear()

self.price\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

self.name\_field.clear()

self.price\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def to\_services\_func(self):

self.cams = services()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/add\_repair\_service.ui", self)

self.setWindowTitle("Додати послугу ремонту")

# Просмотр услуг ремонта

class repairs\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_services\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.update\_button.clicked.connect(self.load\_data\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.edit\_button.clicked.connect(self.edit\_func)

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_repair(code)

self.cams.show()

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def delete\_func(self):

# current\_row = self.tableWidget.currentRow()

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Послуги WHERE Код\_продукту=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.load\_data\_func()

def to\_services\_func(self):

self.cams = services()

self.cams.show()

self.close()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT Код\_продукту, Назва, Вартість FROM Послуги WHERE Тип\_послуги='Ремонт\_одягу'"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])+' ₴'))

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

# print('data loaded ') for debugging

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

# print('err') for debugging

def setupUi(self):

uic.loadUi("uis/repairs\_list.ui", self)

self.setWindowTitle("Список послуг ремонту")

if (self.tableWidget.columnCount() < 3):

self.tableWidget.setColumnCount(3)

self.tableWidget.setColumnCount(3)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(1,319)

self.tableWidget.setColumnWidth(2,270)

self.tableWidget.setHorizontalHeaderLabels(["Код\_продукту","Назва","Вартість"])

self.tableWidget.verticalHeader().setVisible(False)

self.load\_data\_func()

# Редактирование услуги ремонта

class edit\_repair(QMainWindow):

def \_\_init\_\_(self,code):

super().\_\_init\_\_()

self.code = code

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.cancel\_button.clicked.connect(self.to\_repairs\_list\_func)

self.submit\_button.clicked.connect(self.edit\_func)

def to\_repairs\_list\_func(self):

self.close()

def edit\_func(self):

name = self.name\_field.text()

price = self.price\_field.text()

conn = sqlite3.connect('atelie.db')

if name == '' or price == '':

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

row = (name,price,self.code)

# print('DATA =', row) # for debugging

query = '''UPDATE Послуги SET Тип\_послуги='Ремонт\_одягу', Тип\_продукту=NULL, Назва=?, Код\_матеріалу=NULL, Колір=NULL, Фото\_продукту=NULL, Вартість=? WHERE Код\_продукту=?;'''

conn.execute(query, row)

self.name\_field.clear()

self.price\_field.clear()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.close()

def setupUi(self):

uic.loadUi("uis/edit\_repairs.ui", self)

self.setWindowTitle("Редагування запису")

# Создание заказа

class add\_offer(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/plus\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.submit\_button.clicked.connect(self.add\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def load\_data\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

query = '''SELECT ПІБ FROM Клієнти;'''

cur = conn.cursor()

tablerow = 0

for row in cur.execute(query):

self.client\_list.insertItem(tablerow,row[0])

tablerow += 1

query = '''SELECT Назва FROM Послуги;'''

tablerow = 0

for row in cur.execute(query):

self.product\_list.insertItem(tablerow,row[0])

tablerow += 1

query = '''SELECT ПІБ\_працівника FROM Працівники;'''

tablerow = 0

for row in cur.execute(query):

self.worker\_list.insertItem(tablerow,row[0])

tablerow += 1

conn.close()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

conn.close()

def add\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

client\_list\_item = self.client\_list.currentText()

cur = conn.cursor()

row = cur.execute('SELECT Код\_клієнта FROM Клієнти WHERE ПІБ=?',(str(client\_list\_item),))

client\_code = row.fetchall()[0][0]

product\_list\_item = self.product\_list.currentText()

row = cur.execute('SELECT Код\_продукту FROM Послуги WHERE Назва=?',(str(product\_list\_item),))

product\_code = row.fetchall()[0][0]

worker\_list\_item = self.worker\_list.currentText()

row = cur.execute('SELECT Код\_працівника FROM Працівники WHERE ПІБ\_працівника=?',(str(worker\_list\_item),))

worker\_code = row.fetchall()[0][0]

price\_item = self.product\_list.currentText()

row = cur.execute('SELECT Вартість FROM Послуги WHERE Назва=?',(str(price\_item),))

first\_price = row.fetchall()[0][0]

except sqlite3.Error as e:

print(e)

print(e.args)

price = str(int(self.price.text()) + int(first\_price))

if self.payed\_radiobutton.isChecked():

payed = -1

elif self.unpayed\_radiobutton.isChecked():

payed = 0

else:

payed = ''

date\_of\_start = self.start\_date.selectedDate().toString('yyyy-MM-dd')

date\_of\_end = self.end\_date.selectedDate().toString('yyyy-MM-dd')

if client\_code == '' or product\_code == '' or worker\_code == '' or price == '' or date\_of\_start == '' or date\_of\_end == '' or payed == '':

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

# print(f'client-code - {client\_code}\nproduct\_code - {product\_code}\nworker code - {worker\_code}\nprice - {price}\npayed - {payed}\ndate\_of\_start - {date\_of\_start}\ndate\_of\_end - {date\_of\_end}')

row = (client\_code,product\_code,date\_of\_start, date\_of\_end,price,payed,worker\_code)

print('DATA =', row) # for debugging

query = '''insert into Закази (Код\_клієнта, Код\_продукту, Дата\_початку, Дата\_закінчення, Ціна, Оплачено, Працівник)

values (?, ?, ?, ?, ?, ?, ?);'''

conn.execute(query, row)

self.price.clear()

self.payed\_radiobutton.clearFocus()

self.unpayed\_radiobutton.clearFocus()

self.end\_date.clearFocus()

self.start\_date.clearFocus()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

self.price.clear()

self.payed\_radiobutton.clear()

self.unpayed\_radiobutton.clear()

self.end\_date.clearFocus()

self.start\_date.clearFocus()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

def setupUi(self):

uic.loadUi("uis/add\_offer.ui", self)

self.setWindowTitle("Додати заказ")

self.load\_data\_func()

# Просмотр заказов

class offers\_list(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.delete\_button.clicked.connect(self.delete\_func)

self.update\_button.clicked.connect(self.load\_data\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.edit\_button.clicked.connect(self.edit\_func)

self.get\_check\_button.clicked.connect(self.get\_check\_func)

self.change\_paid\_status\_button.clicked.connect(self.change\_paid\_func)

def change\_paid\_func(self):

try:

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

except AttributeError:

QMessageBox.information(self, 'Помилка', 'Не вибрано заказ для зміни статусу оплати')

else:

connection = sqlite3.connect("atelie.db")

with connection:

cur = connection.cursor()

payed = cur.execute('SELECT Оплачено FROM Закази\_подр WHERE Код\_заказу=?',(code,))

payed = payed.fetchall()[0][0]

if payed == 0:

sqlquery = "UPDATE Закази SET Оплачено=-1 WHERE Код\_заказу=?"

elif payed == -1:

sqlquery = "UPDATE Закази SET Оплачено=0 WHERE Код\_заказу=?"

cur.execute(sqlquery,(code,))

connection.commit()

connection.close()

self.load\_data\_func()

def get\_check\_func(self):

try:

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

except AttributeError:

QMessageBox.information(self, 'Помилка', 'Не вибрано заказ для генерації чеку')

else:

connection = sqlite3.connect("atelie.db")

with connection:

cur = connection.cursor()

payed = cur.execute('SELECT Оплачено FROM Закази\_подр WHERE Код\_заказу=?',(code,))

payed = payed.fetchall()[0][0]

if payed == 0:

QMessageBox.information(self, 'Помилка', 'Неможливо видати чек по неоплаченому заказу')

elif payed == -1:

sqlquery = "SELECT Код\_заказу, ПІБ, Назва, ПІБ\_працівника, strftime('%d/%m/%Y', Дата\_початку), strftime('%d/%m/%Y', Дата\_закінчення), Ціна, Оплачено FROM Закази\_подр WHERE Код\_заказу=?"

styles = getSampleStyleSheet()

styles['Normal'].fontName='DejaVuSerif'

styles['Heading1'].fontName='DejaVuSerif'

pdfmetrics.registerFont(TTFont('DejaVuSerif','assets\DejaVuSerif.ttf', 'UTF-8'))

dir = os.getcwd()

dir = os.path.join(dir,'reports')

if not os.path.exists(dir):

os.mkdir(dir)

doc = SimpleDocTemplate(r'reports\check.pdf',

pagesize = A4,

title='Check',

author='hpfk352')

story = [] # словарь документа

story.append(Paragraph('Квитанція про надання послуг',styles['Normal']))

story.append(Paragraph('Ательє "Студія Моди"',styles['Normal']))

story.append(Paragraph('',styles["Normal"]))

row = cur.execute(sqlquery,(code,))

row = row.fetchall()[0]

tblstyle = TableStyle([('FONT', (0, 0), (-1, 1), 'DejaVuSerif', 7),('BOX', (0,0), (-1,-1), 0.25,colors.black)])

t = Table(

[ ['Код заказу','Клієнт','Продукт','Працівник','Початок','Закінчення','Ціна'],

[str(row[0]), str(row[1]), str(row[2]),str(row[3]),str(row[4]),str(row[5]),str(row[6])+' ₴'],

]

)

t.setStyle(tblstyle)

story.append(t)

doc.build(story)

os.startfile(r"reports\check.pdf", "print")

connection.close()

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def delete\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

conn = sqlite3.connect("atelie.db")

try:

with conn:

# print('DATA =', code) # for debugging

query = '''DELETE FROM Закази WHERE Код\_заказу=?;'''

conn.execute(query, (code,) )

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

# self.tableWidget.removeRow(current\_row)

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

msg = QMessageBox()

msg.setWindowTitle("Результат виконання")

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.load\_data\_func()

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT Код\_заказу, ПІБ, Назва, ПІБ\_працівника, strftime('%d/%m/%Y', Дата\_початку), strftime('%d/%m/%Y', Дата\_закінчення), Ціна, Оплачено FROM Закази\_подр"

tablerow = 0

total\_rows\_count = 0

for row in cur.execute(sqlquery):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(sqlquery):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])+' ₴'))

self.check\_box = QCheckBox()

if str(row[7]) == '-1':

self.check\_box.setChecked(True)

elif str(row[7]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 7, self.check\_box)

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def edit\_func(self):

code = self.tableWidget.item(self.tableWidget.currentRow(),0).text()

self.cams = edit\_offer(code)

self.cams.show()

def setupUi(self):

uic.loadUi("uis/offers\_list.ui", self)

self.setWindowTitle("Список замовлень")

self.tableWidget.setColumnCount(8)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(4,80)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setColumnWidth(7,80)

self.tableWidget.setColumnWidth(1,220)

self.tableWidget.setColumnWidth(2,130)

self.tableWidget.setColumnWidth(3,200)

self.tableWidget.setHorizontalHeaderLabels(["№ заказу","Клієнт","Послуга","Працівник","Початок","Закінчення","Ціна","Оплачено"])

self.load\_data\_func()

# Меню заказов

class offers(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/offer\_icon.png"))

self.back\_button.clicked.connect(self.to\_main\_menu\_func)

self.exit\_button.clicked.connect(exit\_func)

self.view\_offers\_button.clicked.connect(self.offers\_list\_func)

self.create\_offer\_button.clicked.connect(self.to\_add\_offer\_func)

self.offers\_by\_day\_button.clicked.connect(self.offers\_by\_day\_func)

self.offers\_by\_month\_button.clicked.connect(self.offers\_by\_month\_func)

self.offers\_by\_months\_button.clicked.connect(self.offers\_by\_months\_func)

self.offers\_by\_period\_button.clicked.connect(self.offers\_by\_period\_func)

def to\_main\_menu\_func(self):

self.cams = main\_menu()

self.cams.show()

self.close()

def offers\_list\_func(self):

self.cams = offers\_list()

self.cams.show()

self.close()

def to\_add\_offer\_func(self):

self.cams = add\_offer()

self.cams.show()

self.close()

def offers\_by\_day\_func(self):

self.cams = select\_day()

self.cams.show()

self.close()

def offers\_by\_month\_func(self):

self.cams = select\_month()

self.cams.show()

self.close()

def offers\_by\_months\_func(self):

self.cams = offers\_per\_month()

self.cams.show()

self.close()

def offers\_by\_period\_func(self):

self.cams = select\_period()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/offers.ui", self)

self.setWindowTitle("Закази")

# Редактирование заказа

class edit\_offer(QMainWindow):

def \_\_init\_\_(self,code):

super().\_\_init\_\_()

self.code = code

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/edit\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_list\_func)

self.submit\_button.clicked.connect(self.edit\_func)

def to\_offers\_list\_func(self):

self.close()

def edit\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

client\_list\_item = self.client\_list.currentText()

cur = conn.cursor()

row = cur.execute('SELECT Код\_клієнта FROM Клієнти WHERE ПІБ=?',(str(client\_list\_item),))

client\_code = row.fetchall()[0][0]

product\_list\_item = self.product\_list.currentText()

row = cur.execute('SELECT Код\_продукту FROM Послуги WHERE Назва=?',(str(product\_list\_item),))

product\_code = row.fetchall()[0][0]

worker\_list\_item = self.worker\_list.currentText()

row = cur.execute('SELECT Код\_працівника FROM Працівники WHERE ПІБ\_працівника=?',(str(worker\_list\_item),))

worker\_code = row.fetchall()[0][0]

price\_item = self.product\_list.currentText()

row = cur.execute('SELECT Вартість FROM Послуги WHERE Назва=?',(str(price\_item),))

first\_price = row.fetchall()[0][0]

except sqlite3.Error as e:

print(e)

print(e.args)

price = str(int(self.price.text()) + int(first\_price))

if self.payed\_radiobutton.isChecked():

payed = -1

elif self.unpayed\_radiobutton.isChecked():

payed = 0

else:

payed = ''

date\_of\_start = self.start\_date.selectedDate().toString('yyyy-MM-dd')

date\_of\_end = self.end\_date.selectedDate().toString('yyyy-MM-dd')

if client\_code == '' or product\_code == '' or worker\_code == '' or price == '' or date\_of\_start == '' or date\_of\_end == '' or payed == '':

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

else:

try:

with conn:

# print(f'client-code - {client\_code}\nproduct\_code - {product\_code}\nworker code - {worker\_code}\nprice - {price}\npayed - {payed}\ndate\_of\_start - {date\_of\_start}\ndate\_of\_end - {date\_of\_end}')

row = (client\_code,product\_code,date\_of\_start, date\_of\_end,price,payed,worker\_code,self.code)

print('DATA =', row) # for debugging

query = '''UPDATE Закази SET Код\_клієнта=?, Код\_продукту=?, Дата\_початку=?, Дата\_закінчення=?, Ціна=?, Оплачено=?, Працівник=? WHERE Код\_заказу=?;'''

conn.execute(query, row)

self.price.clear()

self.payed\_radiobutton.clearFocus()

self.unpayed\_radiobutton.clearFocus()

self.end\_date.clearFocus()

self.start\_date.clearFocus()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/success\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Операція виконана успішно!")

x = msg.exec\_()

except sqlite3.Error as e:

print(e) #for debugging

print(e.args)

self.price.clear()

self.end\_date.clearFocus()

self.start\_date.clearFocus()

msg = QMessageBox()

msg.setWindowIcon(QIcon(u":/newPrefix/assets/error\_icon.png"))

msg.setWindowTitle("Результат виконання")

msg.setText("Виникла помилка, перевірте правильність данних та заповненість всіх полей")

x = msg.exec\_()

conn.commit()

conn.close()

self.close()

def load\_data\_func(self):

conn = sqlite3.connect("atelie.db")

try:

with conn:

query = '''SELECT ПІБ FROM Клієнти;'''

cur = conn.cursor()

tablerow = 0

for row in cur.execute(query):

self.client\_list.insertItem(tablerow,row[0])

tablerow += 1

query = '''SELECT Назва FROM Послуги;'''

tablerow = 0

for row in cur.execute(query):

self.product\_list.insertItem(tablerow,row[0])

tablerow += 1

query = '''SELECT ПІБ\_працівника FROM Працівники;'''

tablerow = 0

for row in cur.execute(query):

self.worker\_list.insertItem(tablerow,row[0])

tablerow += 1

conn.close()

except sqlite3.Error as e:

# print(e) for debugging

# print(e.args)

conn.close()

def setupUi(self):

uic.loadUi("uis/edit\_offer.ui", self)

self.setWindowTitle("Редагування запису")

self.load\_data\_func()

# Заказы за день

class offers\_by\_day(QMainWindow):

def \_\_init\_\_(self, date):

super().\_\_init\_\_()

self.date = date

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_закінчення, Ціна, Оплачено FROM Закази\_подр WHERE Дата\_початку=?",(self.date,)):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_закінчення, Ціна, Оплачено FROM Закази\_подр WHERE Дата\_початку=?",(self.date,)):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])+' ₴'))

self.check\_box = QCheckBox()

if str(row[7]) == '-1':

self.check\_box.setChecked(True)

elif str(row[7]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 7, self.check\_box)

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

def setupUi(self):

uic.loadUi("uis/offers\_by\_day.ui", self)

self.setWindowTitle("Закази за день")

self.tableWidget.setColumnCount(8)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(4,80)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setColumnWidth(7,80)

self.tableWidget.setColumnWidth(1,220)

self.tableWidget.setColumnWidth(2,130)

self.tableWidget.setColumnWidth(3,200)

self.tableWidget.setHorizontalHeaderLabels(["№ заказу","Клієнт","Послуга","Працівник","Початок","Закінчення","Ціна","Оплачено"])

self.load\_data\_func()

# Выбор дня

class select\_day(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/calendar\_icon.png"))

self.submit\_button.clicked.connect(self.to\_offers\_by\_day\_func)

self.cancel\_button.clicked.connect(self.to\_offers\_func)

def to\_offers\_by\_day\_func(self):

date = self.calendarWidget.selectedDate().toString('yyyy-MM-dd')

self.cams = offers\_by\_day(date)

self.cams.show()

self.close()

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/select\_date.ui", self)

self.setWindowTitle("Виберіть день")

# Выбор месяца

class select\_month(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/calendar\_icon.png"))

self.submit\_button.clicked.connect(self.to\_offers\_by\_month\_func)

self.cancel\_button.clicked.connect(self.to\_offers\_func)

def to\_offers\_by\_month\_func(self):

month = self.month\_list.currentText()

month\_dict = {'Січень': '01',

'Лютий': '02',

'Березень': '03',

'Квітень': '04',

'Травень': '05',

'Червень': '06',

'Липень': '07',

'Серпень': '08',

'Вересень': '09',

'Жовтень': '10',

'Листопад': '11',

'Грудень': '12' }

month = month\_dict.get(month,'01')

self.cams = offers\_by\_month(month)

self.cams.show()

self.close()

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/select\_month.ui", self)

self.setWindowTitle("Виберіть місяць")

# Заказы за месяц

class offers\_by\_month(QMainWindow):

def \_\_init\_\_(self, month):

super().\_\_init\_\_()

self.month = month

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_закінчення, Ціна, Оплачено FROM Закази\_подр WHERE strftime('%m', Дата\_початку) = ?",(self.month,)):

total\_rows\_count += 1

if total\_rows\_count == 0:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

elif total\_rows\_count >= 1:

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_закінчення, Ціна, Оплачено FROM Закази\_подр WHERE strftime('%m', Дата\_початку) = ?",(self.month,)):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])+' ₴'))

self.check\_box = QCheckBox()

if str(row[7]) == '-1':

self.check\_box.setChecked(True)

elif str(row[7]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 7, self.check\_box)

tablerow += 1

def setupUi(self):

self.tableWidget.setColumnCount(8)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(4,80)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setColumnWidth(7,80)

self.tableWidget.setColumnWidth(1,220)

self.tableWidget.setColumnWidth(2,130)

self.tableWidget.setColumnWidth(3,200)

self.tableWidget.setHorizontalHeaderLabels(["№ заказу","Клієнт","Послуга","Працівник","Початок","Закінчення","Ціна","Оплачено"])

self.load\_data\_func()

uic.loadUi("uis/offers\_by\_month.ui", self)

self.setWindowTitle("Закази за місяць")

# Заказы по месяцам

class offers\_per\_month(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

query = '''SELECT CASE strftime('%m', Дата\_початку)

WHEN '01' THEN 'Січень'

WHEN '02' THEN 'Лютий'

WHEN '03' THEN 'Березень'

WHEN '04' THEN 'Квітень'

WHEN '05' THEN 'Травень'

WHEN '06' THEN 'Червень'

WHEN '07' THEN 'Липень'

WHEN '08' THEN 'Серпень'

WHEN '09' THEN 'Вересень'

WHEN '10' THEN 'Жовтень'

WHEN '11' THEN 'Листопад'

WHEN '12' THEN 'Грудень'

END AS Місяць, Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_початку, Ціна, Оплачено

from Закази\_подр

ORDER BY Місяць '''

for row in cur.execute(query):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(query):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

# self.tableWidget.item(0,0).setTextColor(QColor(255,0,0)) # установить цвет TODO: надо как то прикрутить к цветам

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])))

self.tableWidget.setItem(tablerow,7,QTableWidgetItem(str(row[7])+' ₴'))

self.check\_box = QCheckBox()

if str(row[8]) == '-1':

self.check\_box.setChecked(True)

elif str(row[8]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 8, self.check\_box)

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

def setupUi(self):

uic.loadUi("uis/offers\_per\_month.ui", self)

self.setWindowTitle("Закази по місяцях")

self.tableWidget.setColumnCount(9)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(4,200)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setColumnWidth(7,80)

self.tableWidget.setColumnWidth(1,80)

self.tableWidget.setColumnWidth(2,180)

self.tableWidget.setColumnWidth(3,150)

self.tableWidget.setHorizontalHeaderLabels(['Місяць',"№ заказу","Клієнт","Послуга","Працівник","Початок","Закінчення","Ціна","Оплачено"])

self.load\_data\_func()

# Выбор периода

class select\_period(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/calendar\_icon.png"))

self.submit\_button.clicked.connect(self.to\_offers\_by\_period\_func)

self.cancel\_button.clicked.connect(self.to\_offers\_func)

def to\_offers\_by\_period\_func(self):

first\_date = self.first\_date.selectedDate().toString('yyyy-MM-dd')

second\_date = self.second\_date.selectedDate().toString('yyy-MM-dd')

self.cams = offers\_by\_period(first\_date, second\_date)

self.cams.show()

self.close()

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def setupUi(self):

uic.loadUi("uis/select\_period.ui", self)

self.setWindowTitle("Виберіть період")

# Заказы за период

class offers\_by\_period(QMainWindow):

def \_\_init\_\_(self, first\_date, second\_date):

super().\_\_init\_\_()

self.first\_date = first\_date

self.second\_date = second\_date

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/list\_icon.png"))

self.back\_button.clicked.connect(self.to\_offers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.print\_button.clicked.connect(self.print\_func)

def to\_offers\_func(self):

self.cams = offers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_початку, Ціна, Оплачено From Закази\_подр WHERE Дата\_початку BETWEEN ? AND ? ",(self.first\_date,self.second\_date,)):

total\_rows\_count += 1

if total\_rows\_count == 0:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

elif total\_rows\_count >= 1:

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute("SELECT Код\_заказу, ПІБ, Назва,ПІБ\_працівника, strftime('%d-%m-%Y',Дата\_початку) as Дата\_початку, strftime('%d-%m-%Y',Дата\_закінчення) as Дата\_початку, Ціна, Оплачено From Закази\_подр WHERE Дата\_початку BETWEEN ? AND ? ",(self.first\_date,self.second\_date,)):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.tableWidget.setItem(tablerow,6,QTableWidgetItem(str(row[6])+' ₴'))

self.check\_box = QCheckBox()

if str(row[7]) == '-1':

self.check\_box.setChecked(True)

elif str(row[7]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 7, self.check\_box)

tablerow += 1

QTimer.singleShot(10000, self.load\_data\_func)

def print\_func(self):

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

sqlquery = "SELECT Код\_заказу, ПІБ, Назва, ПІБ\_працівника, strftime('%d/%m/%Y', Дата\_початку), strftime('%d/%m/%Y', Дата\_закінчення), Ціна, Оплачено FROM Закази\_подр WHERE Код\_заказу =25"

styles = getSampleStyleSheet()

styles['Normal'].fontName='DejaVuSerif'

styles['Heading1'].fontName='DejaVuSerif'

pdfmetrics.registerFont(TTFont('DejaVuSerif','assets\DejaVuSerif.ttf', 'UTF-8'))

doc = SimpleDocTemplate(r'reports\check.pdf',

pagesize = A4,

title='Check',

author='hpfk352')

story = [] # словарь документа

story.append(Paragraph('Квитанція про надання послуг',styles['Normal']))

story.append(Paragraph('Ательє "Студія Моди"',styles['Normal']))

story.append(Paragraph('',styles["Normal"]))

row = cur.execute(sqlquery)

row = row.fetchall()[0]

tblstyle = TableStyle([('FONT', (0, 0), (-1, 1), 'DejaVuSerif', 7),('BOX', (0,0), (-1,-1), 0.25,colors.black)])

t = Table(

[ ['Код заказу','Клієнт','Продукт','Працівник','Початок','Закінчення','Ціна'],

[str(row[0]), str(row[1]), str(row[2]),str(row[3]),str(row[4]),str(row[5]),str(row[6])],

]

)

t.setStyle(tblstyle)

story.append(t)

doc.build(story)

os.startfile(r"reports\check.pdf", "print")

def setupUi(self):

uic.loadUi("uis/offers\_by\_period.ui", self)

self.setWindowTitle("Закази за період")

self.tableWidget.setColumnCount(8)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,80)

self.tableWidget.setColumnWidth(4,80)

self.tableWidget.setColumnWidth(5,80)

self.tableWidget.setColumnWidth(6,80)

self.tableWidget.setColumnWidth(7,80)

self.tableWidget.setColumnWidth(1,220)

self.tableWidget.setColumnWidth(2,130)

self.tableWidget.setColumnWidth(3,200)

self.tableWidget.setHorizontalHeaderLabels(["№ заказу","Клієнт","Послуга","Працівник","Початок","Закінчення","Ціна","Оплачено"])

self.load\_data\_func()

# Отчет по клиентам

class clients\_report(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/report\_icon.png"))

self.back\_button.clicked.connect(self.to\_clients\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.print\_button.clicked.connect(self.print\_func)

def print\_func(self):

connection = sqlite3.connect("atelie.db")

with connection:

cur = connection.cursor()

sqlquery = """select

"Клієнти"."Код\_клієнта",

"Клієнти"."ПІБ",

"Клієнти"."Телефон",

SUM(Закази.Ціна) as Всього\_прибутку

from "Послуги"

join (

"Клієнти"

join "Закази"

on "Клієнти"."Код\_клієнта" = "Закази"."Код\_клієнта"

)

on "Послуги"."Код\_продукту" = "Закази"."Код\_продукту"

group by "Клієнти"."ПІБ"

order by "Всього\_прибутку" DESC"""

styles = getSampleStyleSheet()

styles.add(ParagraphStyle(name='Normal\_CENTER',

parent=styles['Normal'],

fontName='DejaVuSerif',

wordWrap='LTR',

alignment=TA\_CENTER,

fontSize=12,

leading=13,

textColor=colors.black,

borderPadding=0,

leftIndent=0,

rightIndent=0,

spaceAfter=0,

spaceBefore=0,

splitLongWords=True,

spaceShrinkage=0.05,

))

styles['Normal'].fontName='DejaVuSerif'

styles['Heading1'].fontName='DejaVuSerif'

pdfmetrics.registerFont(TTFont('DejaVuSerif','assets\DejaVuSerif.ttf', 'UTF-8'))

dir = os.getcwd()

dir = os.path.join(dir,'reports')

if not os.path.exists(dir):

os.mkdir(dir)

doc = SimpleDocTemplate(r'reports\clients\_report.pdf',

pagesize = A4,

title='Check',

author='hpfk352')

story = []

story.append(Paragraph('Звітність по клієнтам',styles['Normal\_CENTER']))

story.append(Paragraph('Ательє "Студія Моди"',styles['Normal\_CENTER']))

story.append(Paragraph('--------------------',styles["Normal\_CENTER"]))

data = [['Код клієнта','ПІБ клієнта','Телефон','Всього прибутку']]

for row in cur.execute(sqlquery):

data.append([row[0],row[1],row[2],row[3]])

tblstyle = TableStyle([('FONT', (0, 0), (-1, -1), 'DejaVuSerif', 7),('BOX', (0,0), (-1,-1), 0.25,colors.black)])

t = Table(data)

t.setStyle(tblstyle)

story.append(t)

doc.build(story)

os.startfile(r"reports\clients\_report.pdf", "print")

connection.close()

def to\_clients\_func(self):

self.cams = clients()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

query = """select

"Клієнти"."Код\_клієнта",

"Клієнти"."ПІБ",

"Клієнти"."Телефон",

SUM(Закази.Ціна) as Всього\_прибутку

from "Послуги"

join (

"Клієнти"

join "Закази"

on "Клієнти"."Код\_клієнта" = "Закази"."Код\_клієнта"

)

on "Послуги"."Код\_продукту" = "Закази"."Код\_продукту"

group by "Клієнти"."ПІБ"

order by "Всього\_прибутку" DESC"""

for row in cur.execute(query):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(query):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,3,QTableWidgetItem(str(row[3])+' ₴'))

tablerow += 1

def setupUi(self):

uic.loadUi("uis/clients\_zvit.ui", self)

self.setWindowTitle("Звіт по клієнтам")

self.tableWidget.setColumnCount(4)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,120)

self.tableWidget.setColumnWidth(1,250)

self.tableWidget.setColumnWidth(3,150)

self.tableWidget.setColumnWidth(4,150)

self.tableWidget.setHorizontalHeaderLabels(["Код клієнта","ПІБ клієнта","Телефон","Всього прибутку"])

self.load\_data\_func()

# Отчет по работникам

class workers\_report(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon(u":/newPrefix/assets/report\_icon.png"))

self.back\_button.clicked.connect(self.to\_workers\_func)

self.exit\_button.clicked.connect(exit\_func)

self.up\_button.clicked.connect(self.previous\_item\_func)

self.down\_button.clicked.connect(self.next\_item\_func)

self.find\_button.clicked.connect(self.search\_func)

self.tableWidget.clicked.connect(self.currention\_func)

self.print\_button.clicked.connect(self.print\_func)

def print\_func(self):

connection = sqlite3.connect("atelie.db")

with connection:

cur = connection.cursor()

sqlquery = """SELECT Код\_працівника, ПІБ\_працівника, Назва, Оплачено, strftime('%d/%m/%Y', Дата\_початку), strftime('%d/%m/%Y', Дата\_закінчення) FROM Аналіз\_працівників"""

styles = getSampleStyleSheet()

styles.add(ParagraphStyle(name='Normal\_CENTER',

parent=styles['Normal'],

fontName='DejaVuSerif',

wordWrap='LTR',

alignment=TA\_CENTER,

fontSize=12,

leading=13,

textColor=colors.black,

borderPadding=0,

leftIndent=0,

rightIndent=0,

spaceAfter=0,

spaceBefore=0,

splitLongWords=True,

spaceShrinkage=0.05,

))

styles['Normal'].fontName='DejaVuSerif'

styles['Heading1'].fontName='DejaVuSerif'

pdfmetrics.registerFont(TTFont('DejaVuSerif','assets\DejaVuSerif.ttf', 'UTF-8'))

dir = os.getcwd()

dir = os.path.join(dir,'reports')

if not os.path.exists(dir):

os.mkdir(dir)

doc = SimpleDocTemplate(r'reports\workers\_report.pdf',

pagesize = A4,

title='Check',

author='hpfk352')

story = []

story.append(Paragraph('Звітність по працівникам',styles['Normal\_CENTER']))

story.append(Paragraph('Ательє "Студія Моди"',styles['Normal\_CENTER']))

story.append(Paragraph('--------------------',styles["Normal\_CENTER"]))

data = [['Код працівника','ПІБ','Послуга','Оплачено','Початок','Закінчення']]

payed = ''

for row in cur.execute(sqlquery):

if row[3] == -1:

payed = 'Оплачено'

elif row[3] == 0:

payed = 'Не оплачено'

data.append([row[0],row[1],row[2],payed,row[4],row[5]])

tblstyle = TableStyle([('FONT', (0, 0), (-1, -1), 'DejaVuSerif', 7),('BOX', (0,0), (-1,-1), 0.25,colors.black)])

t = Table(data)

t.setStyle(tblstyle)

story.append(t)

doc.build(story)

os.startfile(r"reports\workers\_report.pdf", "print")

connection.close()

def to\_workers\_func(self):

self.cams = workers()

self.cams.show()

self.close()

def previous\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()-1)

def next\_item\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow()+1)

def search\_func(self):

items = self.tableWidget.findItems(self.find\_field.text(), Qt.MatchExactly)

if items:

self.tableWidget.selectRow(items[0].row())

else:

QMessageBox.information(self, 'Search Results', 'Нічого не знайдено. Спробуйте ще раз')

def currention\_func(self):

self.tableWidget.selectRow(self.tableWidget.currentRow())

def load\_data\_func(self):

self.tableWidget.setRowCount(0)

# print('Rows set to 0') for debugging

connection = sqlite3.connect("atelie.db")

cur = connection.cursor()

tablerow = 0

total\_rows\_count = 0

query = """SELECT Код\_працівника, ПІБ\_працівника, Назва, Оплачено, strftime('%d/%m/%Y', Дата\_початку), strftime('%d/%m/%Y', Дата\_закінчення) FROM Аналіз\_працівників"""

for row in cur.execute(query):

total\_rows\_count += 1

self.tableWidget.setRowCount(total\_rows\_count)

# print(f'rows set to {total\_rows\_count}') for debugging

for row in cur.execute(query):

# print(row) for debugging

self.tableWidget.setItem(tablerow,0,QTableWidgetItem(str(row[0])))

self.tableWidget.setItem(tablerow,1,QTableWidgetItem(str(row[1])))

self.tableWidget.setItem(tablerow,2,QTableWidgetItem(str(row[2])))

self.tableWidget.setItem(tablerow,4,QTableWidgetItem(str(row[4])))

self.tableWidget.setItem(tablerow,5,QTableWidgetItem(str(row[5])))

self.check\_box = QCheckBox()

if str(row[3]) == '-1':

self.check\_box.setChecked(True)

elif str(row[3]) == '0':

self.check\_box.setChecked(False)

self.check\_box.setDisabled(True)

self.tableWidget.setCellWidget(tablerow, 3, self.check\_box)

tablerow += 1

def setupUi(self):

self.tableWidget.setColumnCount(6)

self.tableWidget.verticalHeader().setVisible(False)

self.tableWidget.setColumnWidth(0,120)

self.tableWidget.setColumnWidth(1,230)

self.tableWidget.setColumnWidth(2,150)

self.tableWidget.setColumnWidth(3,80)

self.tableWidget.setColumnWidth(4,100)

self.tableWidget.setColumnWidth(5,100)

self.tableWidget.setHorizontalHeaderLabels(["Код працівника","ПІБ","Послуга","Оплачено","Початок","Закінчення"])

self.load\_data\_func()

uic.loadUi("uis/workers\_report.ui", self)

self.setWindowTitle("Звіт по працівникам")

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

widget = start()

widget.show()

try:

sys.exit(app.exec\_())

except:

print('exiting..')

Додаток В

Роздрук результатів роботи програми

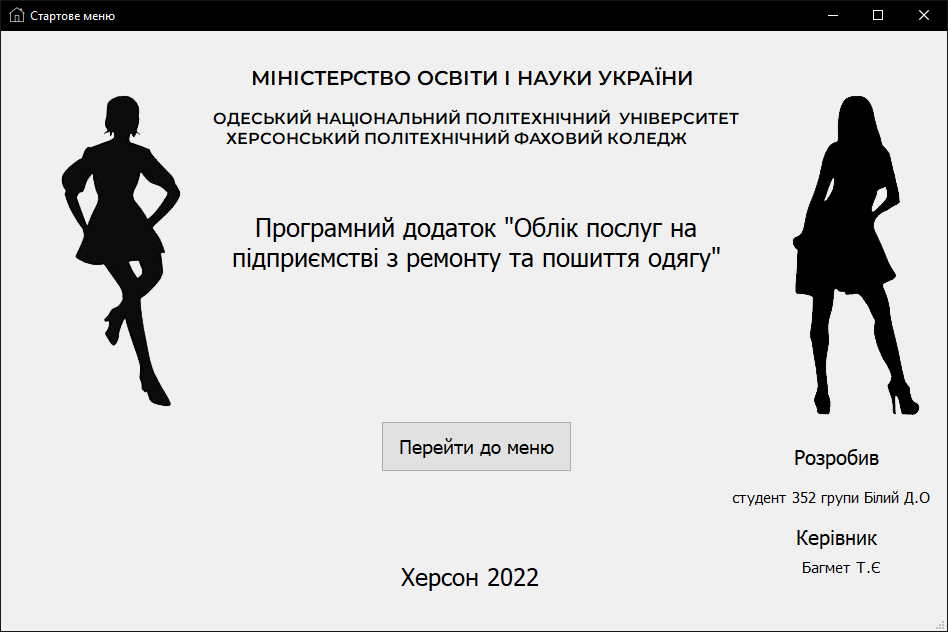


Рисунок А.1 – Стартова форма

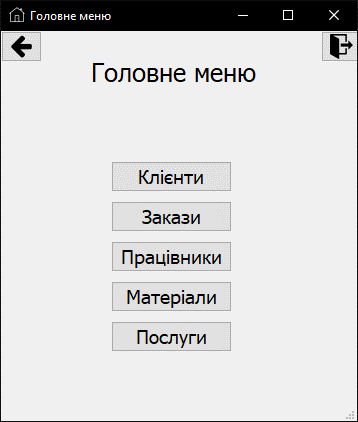
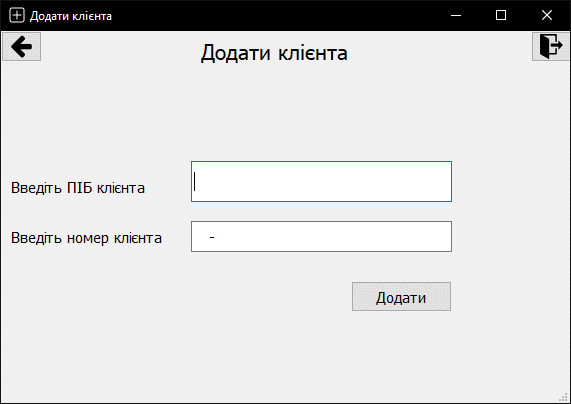


Рисунок А.2 – Головна форма

Рисунок А.3 – Форма додавання клієнта

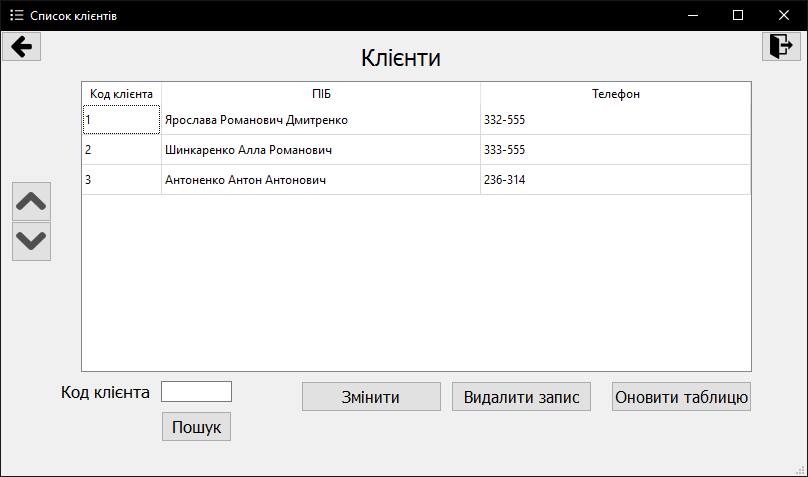


Рисунок А.4 – Форма перегляду списку клієнтів

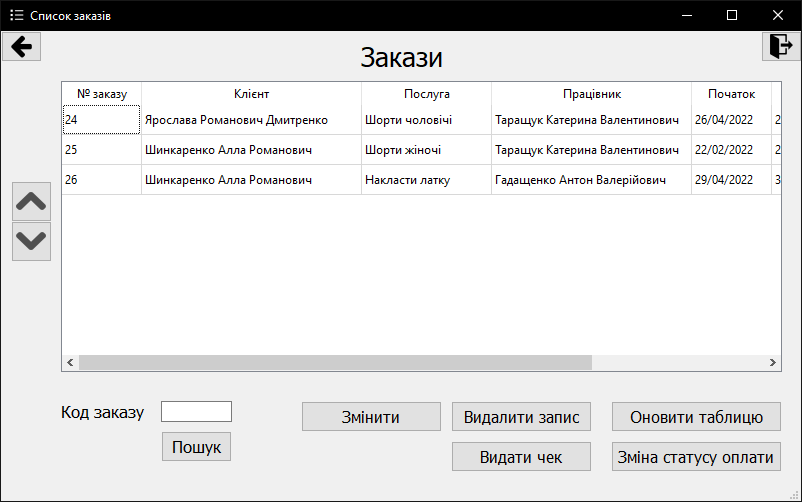


Рисунок А.5 – Форма перегляду замовлень

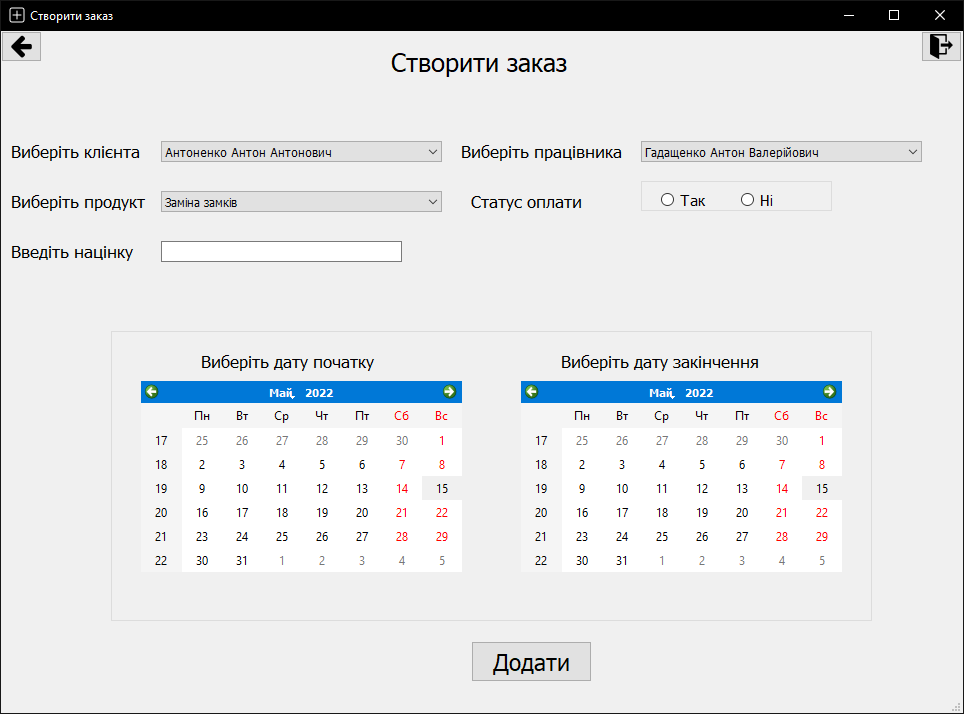
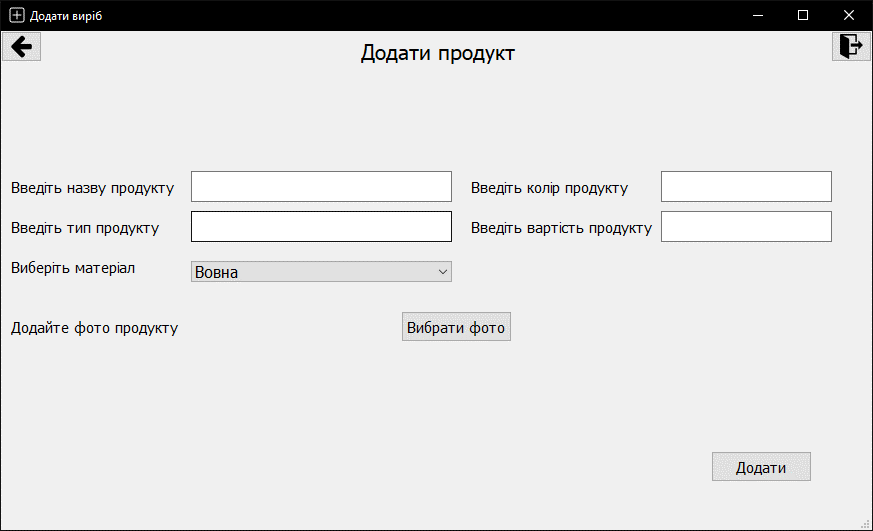


Рисунок А.6 – Форма створення заказу

Рисунок А.7 – Форма додавання виробу

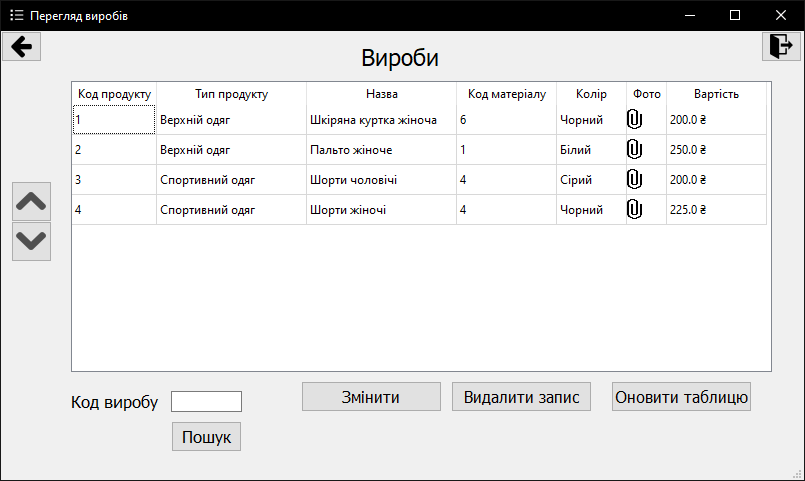


Рисунок А.8 – Форма перегляду виробів

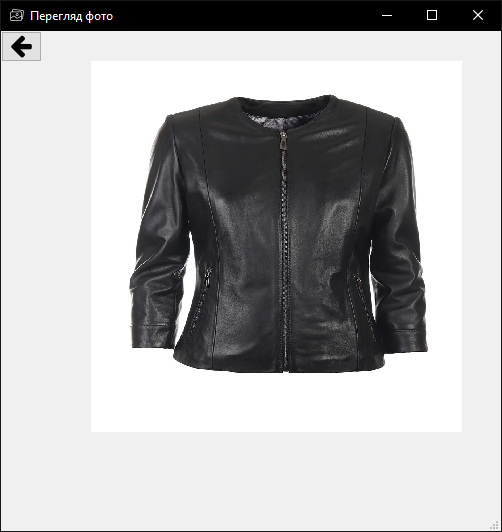
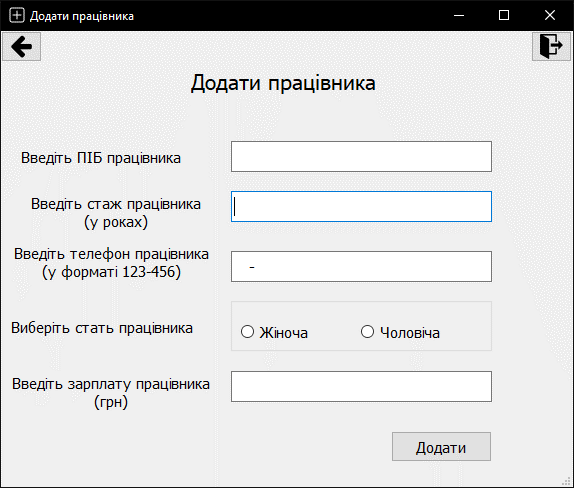


Рисунок А.9 – Перегляд фото виробу

Рисунок А.10 – Форма додавання робітника