

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних систем

Кафедра інженерії програмного забезпечення

КУРСОВИЙ ПРОЕКТ

з «Конструювання програмного забезпечення» та «Бази даних»
на тему: "Розробка інформаційних системи для пункту ксерокопії"

Студента Воробйов В. П.

групи АС-221

спеціальності 121 «Інженерія програмного забезпечення»

Керівник Кунгурцев О.Б.

АНОТАЦІЯ

Ця курсова робота присвячена проектуванню та розробці системи автоматизації пункту друкарні з використанням сучасних технологій та програмного забезпечення. Метою цього проекту є оптимізація процесів, пов'язаних із збором інформації, замовленнями, та поліпшення продуктивності друкарні. У роботі детально розглядаються наступні аспекти: аналіз поточного стану замовлення автоматизації, проектування системи автоматизації, розробка програмного забезпечення, тестування, валідація проекту та висновки.

До аналізу поточного стану входить: оцінка існуючих процесів, обладнання та програмного забезпечення, ідентифікуються основні проблеми та обмеження. До проектування системи автоматизації: визначаються вимоги до системи, розроблюється архітектура та концепція роботи. До розробки програмного забезпечення: створення програмна частина системи, яка включає в себе представлення потрібної інформації. До тестування та валідація: перевірка роботи системи в реальних умовах та визначення її можливості та обмеження. До висновку відноситься узагальнення результатів проекту.

ЗМІСТ

АНОТАЦІЯ.....	2
ЗМІСТ	3
1 ВСТУП.....	7
1.1 Суть розробки:.....	7
1.2 Цілі розробки.....	7
1.3 Задачі проектування.....	8
2 ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ	9
2.1 Документ «Бачення»	9
2.1.1 Введення.....	9
2.1.1.2 Мета.	9
2.1.1.3 Контекст.....	9
2.1.1.4 Визначення, акроніми та скорочення.....	9
2.1.1.5 Посилання.	10
2.1.1.6 Короткий зміст.	10
2.1.2 Позичування	10
2.1.2.1 Ділові переваги.....	10
2.1.2.2 Визначення проблеми.	10
2.1.2.3 Визначення позиції виробу.	13
2.1.3 Опис користувачів.....	14
2.1.3.1 Відомості про користувачів.	14
2.1.3.2 Користувальницька середа.....	14
2.1.3.3 Профілі користувачів.....	15
2.1.3.4 Ключові потреби користувачів.....	16
2.1.4 Короткий огляд виробу.....	16
2.1.4.1 Контекст використання системи.	17
2.1.4.2 Зведення можливостей.....	17
2.1.4.3 Припущення і залежність.....	18
2.1.5 Можливості продукту.....	18
2.1.5.1 Введення даних.....	18
2.1.5.2 Автоматизація.	18

2.1.5.3 Виведення даних.	18
2.1.6 Обмеження.....	19
2.1.7 Показник якості	21
2.1.7.1 Застосування:.....	21
2.1.7.2 Надійність:	21
2.1.8 Інші вимоги до виробу	21
2.1.8.1 Застосовувані стандарти.....	22
2.1.8.2 Системні вимоги.	22
2.1.8.3 Експлуатаційні вимоги.....	22
2.1.9 Вимоги до документації.....	22
2.1.9.1 Керівництво користувача.	22
2.1.9.2 Довідка.	23
2.1.9.3 Керівництва встановлення і конфігурування, файл Read Me.	23
2.2 Прецеденти	23
2.2.1 Опис прецедентів	24
2.2.1.1 Замовлення послуги	24
2.2.1.2 Отримання послуги.....	26
2.2.1.3 Виконання послуги.....	27
3 МОДЕЛЬ КОНЦЕПТУАЛЬНИХ КЛАСІВ	29
3.1 опис прецедентів з назвою можливих класів для реалізації кожного пункту сценарія.	29
3.2 діаграма концептуальних класів для першого варіанта використання	31
3.3 діаграма концептуальних класів для першого і другого варіанта використання	32
3.4 загальна діаграма концептуальних класів:.....	33
4 ДІАГРАМИ ВЗАЄМОДІЇ.....	35
4.1 діаграма системних операцій для «важливого» прецедента:	35
4.2 обґрунтування вибору класу – контролер.	36
4.3 діаграми взаємодії для кожного пункту сценарію «важливого» прецеденту:	36
4.3.1 Проектне рішення newOrder	36
4.3.2 Проектне рішення addType.....	36

4.3.3 Проектне рішення getTerm	37
4.3.4 Проектне рішення getPrice.....	37
4.3.5 Проектне рішення makePayment	37
4.4 обґрунтування прийнятого розподілу обов’язків з посиланням на шаблони проектування:	38
5 МОДЕЛЬ ДАНИХ	39
5.1 Концептуальна модель даних.....	39
5.2 Реляційна модель даних	40
5.3 Обґрунтування вибору первинних ключів.....	41
6 ДІАГРАМИ ПРОГРАМНИХ КЛАСІВ	42
6.1 Специфікація програмного класу Money.....	42
6.2 Специфікація програмного класу Operator	42
6.3 Специфікація програмного класу Order.....	43
6.4 Специфікація програмного класу OrderIDAndOddMoney	43
6.5 Специфікація програмного класу PickUpStation	43
6.6 Специфікація програмного класу Register	44
6.7 Специфікація програмного класу Term.....	45
6.8 Специфікація програмного класу Ticket	45
6.9 Специфікація програмного класу TicketList.....	46
6.10 Специфікація програмного класу Type.....	46
6.11 Специфікація програмного класу TypeItem.....	47
6.12 Специфікація програмного класу TypeList.....	47
6.12 Специфікація програмного класу OrderList.....	47
6.12 Специфікація програмного класу OrderViewAndTypesViews	48
6.13 обґрунтування прийнятих рішень, щодо визначення методів і атрибутів класів; Діаграма програмних класів.	48
6.14 обґрунтування виявлених відношень між класами:	51
7 ПРОГРАМНІ КЛАСИ	53
8 ЗАПИТИ ДО БАЗИ ДАНИХ	54
9 ТЕСТУВАННЯ	66
ВИСНОВКИ	71
СПИСОК ЛІТЕРАТУРИ	72
ДОДАТОК А КОД МОУДЛЯ	73

ДОДАТОК Б КОД КОНТРОЛЕРІВ, ТОЧКУ ВХОДУ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ТА MODUL-INFO.....	111
ДОДАТОК В СТРУКТУРА ФАЙЛІВ КОДУ ПРОГРАМИ.....	183
ДОДАТОК Г КОД КОНФІГУРАЦІЇ ДЛЯ MAVEN	185
ДОДАТОК Д ІНСТРУКЦІЯ КОРИСТУВАЧА	190

1 ВСТУП

Назва проекту: Автоматизація для пункту ксерокопії.

Потрібно розробити автоматизацію, що дозволяє автоматизувати роботу бухгалтера. У функції бухгалтера входить: внесення, зміна, розрахунок даних о прибутку пункту і розрахунку відсотка податку і проценту для банку. Основна його мета - давати уявлення о фінансах компанії, її витратах і доходів.

1 1.1 Суть розробки:

Пункт ксерокопії “ТОВ рога й копита” працює у центрі села. 1 продавець послуг здійснює прийом замовлень та друк. 1 бухгалтер здійснює розрахунки всього підприємства. 1 оператор технічної підтримки ксерокопіювальної машини здійснює покупку матеріалів для друку. Затрати на повноцінну роботу бухгалтера дуже високі, а почерк дуже важкий для розуміння. Помилки при розрахунку бюджету приводить до проблем у праці пункту. Потрібно зменшити затрати за допомогою програмного забезпечення

1.2 Цілі розробки

Метою курсового проекту є поглиблення та закріплення знань, одержаних при вивченні дисциплін «Конструювання програмного забезпечення» та «Бази даних», та набуття практичних навичок у проходженні усіх етапів конструювання об'єктно-орієнтованих програмних модулів.

Під час виконання курсового проекту студенти проходять повний цикл розробки модуля програмної системи. Програмний модуль, що розробляється, має задовольняти всім вимогам, які сформульовані у документі «Бачення».

Завдання на розробку вже отримані студентами у процесі вивчення дисципліни «Аналіз вимог до ПЗ». У рамках даного курсового проекту потрібно уточнити завдання до конкретного програмного модуля, який підлягає проектуванню.

Потрібно розробити автоматизацію, що дозволяє автоматизувати роботу бухгалтера. У функції бухгалтера входить: внесення, зміна, розрахунок даних о прибутку пункту і розрахунку відсотка податку і проценту для банку. Основна його мета - давати уявлення о фінансах компанії, її витратах і доходів.

1.3 Задачі проектування

Потрібно спроектувати проект з урахуванням подальшого розширення програмного продукту, створення на надійному та корпоративному рівні, тобто з можливістю масштабувати

2 ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

З приводу цілей задач та сенсу розробки слідує створення документа «Бачення».

2.1 Документ «Бачення»

Для створення уяви приведемо наступне.

2.1.1 Введення

За слідуючими пунктами буде створене початкове уявлення о проєкті.

2.1.1.2 Мета.

Мета створення цього документа полягає в тому, щоб зібрати, проаналізувати та визначити високорівневі потреби й можливості пункту. Документ наголошує можливості, необхідних співвласникам і цільовим користувачам, і на те, чому ці потреби існують. Подробиці того, як автоматизація пункту виконує ці потреби, будуть деталізовані в прецедентах і додаткових специфікаціях.

2.1.1.3 Контекст.

Цей документ розробляється в рамках проєкту автоматизації діяльності пункту друку “ТОВ рога й копита”.

2.1.1.4 Визначення, акроніми та скорочення.

Основні визначення наведені в документі «Глосарій»

2.1.1.5 Посилання.

Бачення базується на слідуючих документах: “Компл_завд_самостіної_роботи”, “Му_практ_Треб_2022”, “Практична робота 1”, “Практична робота 2”, “Практична робота 3”, “Практична робота 4” від 2022 та 2023 року відповідно.

2.1.1.6 Короткий зміст.

Документ описує високорівневі вимоги до табличного процесора пункту ксерокопії. Вказані основні ділові переваги розглянутого в Баченні рішення, сформульовані ключові проблеми та способи їх розв'язання, наведено характеристики користувачів системи, можливості системи, обмеження, показники якості та інші вимоги до продукту.

2.1.2 Позиціювання

Позиціонування дає можливість зглянути на проблему щиро.

2.1.2.1 Ділові переваги.

В цей час рахування прибутку здійснюється в пункті ксерокопіювання “ТОВ рога й копита” за допомогою паперу. У порівнянні з наявним, нове рішення дозволить забезпечити більш зручний режим доступу зацікавлених осіб до інформації, підвищити швидкодію, забезпечити надійне зберігання даних і повне охоплення функцій, що підлягають автоматизації.

2.1.2.2 Визначення проблеми.

У таблицях 1.1, 1.2, 1.3, 1.4 визначені основні проблеми замовника.

Таблиця 1.1 - Проблема 1

Елемент	Опис
Проблема	Витрати великих грошей та часу на ведення паперового журналу відносно інших подібних організацій.
Впливає на...	Бухгалтера, продавця
Результатом чого є...	Менший грошовий пул для розвитку бізнесу відносно автоматизованих конкурентів. Нестійкість бумажного пера бухгалтера.
Виграш від...	Використання табличного процесора.
Може складатися з	Можливості дешево і швидко передавати данні бухгалтерії. Можливо створити легко копію даних. Автоматизувати розрахунки, а також розрахування під час продажу. Витрачати гроші на електропостачання буде менше ніж на папір та розрахунки бухгалтера.

Таблиця 1.2 - Проблема 2

Елемент	Опис
Проблема	Витрати великих грошей на розрахунки бухгалтером заробітних плат робітників та долі інвесторів відносно інших подібних організацій.
Впливає на...	Бухгалтера, продавця, оператора техніки, прибиральника, менеджера молодшої ланки,

	менеджера середньої ланки, менеджера старшої ланки, директора, помічника директора, інвесторів.
Результатом чого є...	Менший грошовий пул для розвитку бізнесу відносно конкурентів та заробітний борг.
Виграш від...	Використання автоматизації розрахунків у табличного процесора.
Може складатися з	Автоматизування розрахунків, а також розрахування під час продажу. Представити наглядно зібранні й розраховані данні.

Таблиця 1.3 - Проблема 3

Елемент	Опис
Проблема	Витрати великих грошей на розрахунки споживання електроенергії відносно інших подібних організацій.
Впливає на...	Бухгалтера.
Результатом чого є...	Менший грошовий пул для розвитку бізнесу відносно автоматизованих конкурентів. Трата грошей через не велику точність розрахунків.
Виграш від...	Більш точних розрахунків електропостачання.
Може складатися з	Автоматизування розрахунків, а також розрахування під час продажу. Менша витрата грошей на послуги бухгалтера та

	електропостачання.
--	--------------------

Таблиця 1.4 - Проблема 4

Елемент	Опис
Проблема	Витрати великих грошей на розрахунки й планування придбання матеріалів відносно інших подібних організацій.
Впливає на...	Бухгалтера, оператора техніки.
Результатом чого є...	Менший грошовий пул для розвитку бізнесу відносно автоматизованих конкурентів. Трата грошей через не велику точність та повільність розрахунків.
Виграш від...	Використання автоматизації розрахунків через програмну систему. Більш точних розрахунків попит, цін, строк постачання матеріалів. Автоматизування розрахунків, а також розрахування під час продажу.
Може складатися з...	Менша трата грошей на послуги бухгалтера та неточність.

З таблиць 1.1, 1.2, 1.3 і 1.4 можливо знайти особисті проблеми і визначити головні.

2.1.2.3 Визначення позиції виробу.

На таблиці 2.1 зображена підсумкова позиція щодо виробу.

Таблиця 2.1 - Позиція вибору

Для	Пункт ксерокопії “ТОВ рога й копита”
якої	Потрібно оптимізувати процес ведення бухгалтерії
(Назва продукту)	«Автоматизація для пункту ксерокопії»
який	Заснований на промислової СУБД і високонадійний
На відміну від	Існуючого механізму на основі паперу

Через це, ми можемо визначити послідує положення.

2.1.3 Опис користувачів

Опис користувачів надасть визначити їх потреби.

2.1.3.1 Відомості про користувачів.

У системі існують три основних користувачі: продавець, бухгалтер, оператор техпідтримки. Продавець - продає послуги ксерокопіювання та друкує. Бухгалтер - розраховує фінанси у компанії. Оператор - займається підтримкою ксерокопіювального апарату та замовляє матеріали для ксерокопіювання.

2.1.3.2 Користувальницька середовище.

В цей час на підприємстві є десять бухгалтерів, десять продавців і десять операторів. Збільшення персоналу не планується. У цей час розрахунок відбувається на папері за 2 години.

Система буде працювати на платформі IBM PC. Операційна система: Microsoft Windows 10.

2.1.3.3 Профілі користувачів.

З таблиці 3.1 зібрано профіль бухгалтера, з якого слідує головне його положення серед інших працівників, таблиця 3.2 і 3.3, для табличного процесора.

Таблиця 3.1 - Профіль бухгалтера

Типовий представник	Бухгалтер
Опис	Користувач системи, наділений правами на читання інформації, занесення даних про фінанси та використовувати формули для автоматизації. Розраховує заробітну плату працівникам.
Тип	Користувач
Відповідальності	Вводить дані про фінанси компанії. Розраховує також податки й прибутки.
Критерій успіху	Збільшення ефективності, безпеки даних, зменшення помилок при рахуванні, зменшення потреби у бухгалтері

Таблиця 3.2 - Профіль продавця

Типовий представник	Продавець
Опис	Заносить дані у фінансовий журнал щодо продажу послуг. Продає послуги ксерокопіювання.

Тип	Користувач
Відповідальності	Заносить дані у фінансовий журнал щодо продажу послуг.
Критерій успіху	Збільшення ефективності, безпеки даних, зменшення помилок при рахуванні, зменшення часу для внесення даних.

Таблиця 3.3 - Профіль оператора

Типовий представник	Оператор
Опис	Заносить дані у фінансовий журнал щодо витрат на матеріали.
Тип	Користувач
Відповідальності	Вводить дані про трату грошей на матеріали.
Критерій успіху	Збільшення ефективності, безпеки даних, зменшення помилок при рахуванні, зменшення часу для внесення даних.

2.1.3.4 Ключові потреби користувачів.

Бухгалтерія витрачає велику кількість часу на розрахунки фінансів. Бухгалтерія витрачає велику кількість часу через втрату документацій на папері та через помилки у розрахунків. Підприємство потребує у табличному процесорі, який збільшує ефективність і прискорює роботу.

2.1.4 Короткий огляд виробу

Короткий огляд дасть представити певну картину розуміння майбутньої програми.

2.1.4.1 Контекст використання системи.

Система є закінченою незалежною розробкою. Комунікації – на рівні доступу до загальної бази даних.

2.1.4.2 Зведення можливостей.

З таблиці 4.1 приведені можливості програми, які будуть задовольняти потреби замовника.

Таблиця 4.1 - Можливості програми

Вигоди замовника	Підтримують можливості
Спрощення роботи бухгалтера	Автоматичні функції; управління функціями; коригування розрахунків.
Прискорення обігу інформації	Система дозволить прискорити процес отримання необхідної інформації, оптимізує взаємодію продавця та бухгалтера, оператора, менеджерами, податковій службі та банку.
Формування єдиної бази для планування та аналізу	Всі зацікавлені користувачі зі своїх робочих місць мають доступ до інформації з заробітних плат та фінансів компанії; накопичені в базі дані дозволять здійснити аналіз прибутків.

Визначення можливостей програми дасть уявні вимоги.

2.1.4.3 Припущення і залежність.

Система буде використовуватися на територіально зосередженому (без зовнішніх філій) підприємстві.

У разі змін у формах документів ТП повинна зазнати не великі зміни (потрібно буде модифікувати звітні форми).

У випадку придбання або розробки інформаційних систем, що автоматизують суміжні ділянки (маркетинг, склад готової продукції), буде необхідно розробити відповідні засоби імпорту / експорту інформації.

2.1.5 Можливості продукту

Визначення можливостей продукту дасть явне уявлення вимог до реалізації функціоналу програми.

2.1.5.1 Введення даних.

Введення даних у відповідну секцію і таблицю

2.1.5.2 Автоматизація.

Створення функцій, які автоматично розраховують обрані клітини й виводять результат у клітину з формулою.

2.1.5.3 Виведення даних.

Дані будуть виводитись о той самій таблиці, де вносяться. Таблицю можливо буде видрукувати.

2.1.6 Обмеження

На таблиці 5.1 зображено обмеження майбутньої програми і їх опис.

Таблиця 5.1 - Обмеження програми

Джерело	Обмеження	Пояснення
Економічний	Використання відкритого програмного забезпечення.	Перенавантажений продукт буде тільки тратити гроші, а за використання за основою відкритого програмного забезпечення не потребує для використання грошей, крім послуг підтримки.
Політичний	Активне використання демонстрації даних таблиці через друкування та фізичне передавання даних.	Через не поширеність між деяких робітників і зацікавлених осіб ПК, треба представити інформацію іншими шляхами, наприклад, через надання

		результату розрахунків у вигляді друкованого документа.
Технічний	Розробка системи зі строгою об'єктно-орієнтованою методологією, використанням вільною СУБД та тяжким клієнтом. Програма повинна бути збудована на вільній ліцензії.	Програма повинна бути збудована на вільній ліцензії, але саме розробка може використовувати програми іншої ліцензії, а сама кінцева програма пропрієтарним власником якої буде замовник. Під тяжким клієнтом мається на увазі що весь розрахунок буде відбуватись саме на комп'ютері клієнта.
Системний	Мультиплатформність для операційних систем Windows з 2 Гб оперативної пам'яті, 5 Гб вільного дискового простору, двоядерний процесор з тактовою частотою 2 ГГц.	Пристрій продавця та бухгалтера мають різні операційні системи, також планується не скоро зміна системного оснащення.

Експлуатаційн ий	Обмеження до використання пропрієтарного програмного забезпечення. Тільки вільне програмне забезпечення.
Графік і ресурси	Обмеженням часу є 3 місяці. Бюджет достатній тільки на заробітну плату розробників ПЗ.

Знання обмежень дасть розуміння щодо вимог.

2.1.7 Показник якості

Показник якості дасть нам розуміння і певні потреби до потрібній якості продукту.

2.1.7.1 Застосування:

- Час, необхідний для навчання звичайних користувачів – 3 робочих дні (24 години), для навчання просунутих користувачів – 1 робочий день (8 годин).
- Час відгуку для типових завдань – не більше 5 секунд, для складних завдань – не більше 20 секунд.

2.1.7.2 Надійність:

- Доступність – час, що витрачається на обслуговування системи не повинно перевищувати 5% від загального часу роботи.
- Середній час безвідмовної роботи – 1 робочий день.
- Максимальна норма помилок або дефектів – 1 помилка на тисяче рядків коду.

2.1.8 Інші вимоги до виробу

Також існують інші вимоги, які потрібно рішити перед початком розробки програмного забезпечення.

2.1.8.1 Застосовувані стандарти.

Система повинна відповідати всім стандартам інтерфейсу користувача Microsoft Windows.

2.1.8.2 Системні вимоги.

Мінімальні системні вимоги:

- 8 Gb оперативної пам'яті
- 40 Gb вільного дискового простору
- процесор з тактовою частотою не нижче 2.1 GHz
- Операційна система Windows 10 і вище.

2.1.8.3 Експлуатаційні вимоги.

Система повинна бути здатна підтримувати мінімум 1 одночасно працюючих користувачів на одному комп'ютері. Комп'ютер повинен бути увімкненим.

2.1.9 Вимоги до документації

Для подальшої розробки й розуміння можливого застосування використовується документація.

2.1.9.1 Керівництво користувача.

У системі повинні бути представлені Керівництва користувачів (за типами користувачів). Вони повинні містити розшифровку всіх використовуваних термінів, опису основних варіантів використання, включаючи альтернативні сценарії, а також докладний огляд інтерфейсу програми.

2.1.9.2 Довідка.

Довідка необхідна для розв'язання питань, що виникли під час роботи. Довідка повинна містити максимально повну і детальну інформацію по роботі системи.

2.1.9.3 Керівництва встановлення і конфігурування, файл Read Me.

Система повинна мати керівництво по установці в файлі ReadMe.txt, який повинен додаватися до системи. Файл ReadMe.txt повинен містити докладну інструкцію з встановлення даної системи, щоб у разі необхідності користувач зміг виробити установку самостійно, без допомоги адміністратора.

2.2 Прецеденти

З приводу документа бачення створюються наступні прецеденти
Діаграма прецедентів на рисунку 1.1.



Рисунок 1.1 - UML діаграма прецедентів.

На рисунку 1.1 зображено взаємодія користувачів з програмою і взаємодія програми з користувачами.

2.2.1 Опис прецедентів

2.2.1.1 Замовлення послуги

Основна діюча особа: продавець.

Учасники та інтереси:

Касир – продати послугу.

Клієнт – отримати послугу.

Директор, банк, податкова служба – отримати фінансовий протокол від продажі послуги.

Передумова: Працюючий у даний момент касир.

Мінімальна гарантія: Клієнт звертається за послугою.

Гарантія успіху: Клієнт отримує квитанцію за замовленням.

Тригер: Клієнт звертається за послугою.

Основний сценарій:

1. Клієнт звертається за послугою у зазначений пункт. Касир створює нове замовлення у системі. Система фіксує пункт, де створюється замовлення.

2. Касир запитує тип послуги. Клієнт говорить тип послуги. Касир вводить у систему сказаний тип послуги. Система підтверджує і фіксує.

3. Касир запитує кількість надаваної послуги. Клієнт говорить кількість потрібної послуги. Касир вводить у систему сказану кількість. Система підтверджує і фіксує.

4. Касир отримує з системи приблизний час виконання послуги. Касир говорить клієнту цей час.

5. Касир запитує у системи вартість надаваної послуги. Касир говорить клієнту отриману з системи вартість послуги. Клієнт згоден з вартістю.

6. Клієнт дає відповідну кількість грошей, ПІБ, контактний телефон. Касир вводить усе це у систему. Система перевіряє і фіксує замовлення, вибирає оператора, генерує квитанцію і змінює у відповідний стан замовлення.

Розширення:

1а. Відмова клієнта замовляти послугу.

1а1. Клієнт відмовляється від послуги після створення замовлення. Касир скасовує замовлення у системі.

1а2. Система фіксує скасування замовлення.

2а. Помилка при вводі інформації.

2а1. Касир допускає помилку при введенні типу послуги, кількості або вартості. Система повідомляє про помилку та запитує коректну інформацію.

2а2. Касир виправляє помилку та повторно вводить інформацію. Система підтверджує правильність даних.

3а. Несподівана помилка при вказі кількості послуг.

3а1. Касир допускає помилку при введенні кількості надаваної послуги. Система повідомляє про помилку та запитує коректну кількість послуг.

3а2. Касир виправляє помилку та повторно вводить кількість послуг. Система підтверджує правильність даних.

4а. Клієнт не погоджується з часом виконання послуги.

4а1. Система надає приблизний час виконання послуги, касир говорить клієнту, цей приблизний час, але клієнт не згоден з цим часом.

4а2. Касир попереджає клієнта, що це приблизний час, та запитує, чи він готовий прийняти послугу в інший час.

5а. Клієнт відмовляється від вартості послуги:

5а1. Система надає вартість послуги, касир говорить її клієнту, але клієнт не погоджується з ціною.

5a2. Касир може спробувати переговорити щодо ціни або запропонувати альтернативні варіанти послуги з іншою вартістю.

5a3. Якщо клієнт все одно відмовляється, замовлення відмінюють.

6a. Касир вносить неправильну інформацію:

6a1. Касир вводить інформацію у систему. Система виводить помилку вводу.

6a2. Касир просить Клієнта надати ще раз дані. Касир вводить їх у систему.

6b3. Якщо система знову виводить помилку, замовлення відмінюється, гроші повертаються.

2.2.1.2 Отримання послуги

Основна дієва особа: Продавець.

Учасники та інтереси:

Касир – видати послугу.

Клієнт – отримати послугу.

Оператор – створити послугу.

Менеджер – вирішувати не задокументовані питання.

Передумова: Працюючий у даний момент касир, існуюче замовлення, створена послуга.

Мінімальна гарантія: Клієнт повертається за послугою до каси.

Гарантія успіху: Клієнт отримує замовлену послугу.

Тригер: Замовлена послуга створена.

Основний сценарій:

1. Система повідомляє касира о готовності замовлення. Касир повідомляє клієнту о готовності замовлення.

2. Клієнт повертається до касира та дає квитанцію. Касир вводить у систему номер квитанції. Система перевіряє і Касир отримує з систему інформацію о статусі замовлення. Якщо замовлення готово, то касир

підтверджує надання замовлення у системі та передає результат послуги клієнту.

Розширення:

1a. Система повідомляє о неможливості завершити замовлення.

1a1. Система повідомляє Клієнта о неможливості завершити замовлення та потребує прийти до каси.

1a2. Клієнт підходить до каси, дає свою квитанцію. Касир перевіряє через систему статус замовлення. Якщо статус підтверджує неможливість виконати замовлення, то Касир оформляє повернення грошей через систему та повертає гроші клієнту з каси, за замовлення.

2a. Квитанція не дійсна.

2a1. Клієнт надає квитанцію. Касир перевіряє її через систему. Касир отримує з систему інформацію о недійсності замовлення.

2a2. Касир говорить Клієнту о недійсності замовлення та пропонує звернутись до менеджера для вирішення питання.

2.2.1.3 Виконання послуги

Основна дієва особа: Оператор.

Учасники та інтереси:

Оператор – створити послугу.

Менеджер – вирішувати не задокументовані питання.

Передумова: Наявність у пункті оператора на даний момент, існуюче не закінчене замовлення.

Мінімальна гарантія: Оператор побачить нові замовлення.

Гарантія успіху: Оператор створює замовлену послугу.

Тригер: Отримання інформації о новому замовленні.

Основний сценарій:

1. Система повідомляє о наявності невиконаного замовлення для конкретного оператора. Оператори дивяться і відповідний оператор запитує у системі більше додаткової інформації щодо замовлення.

2. Система надає додаткову інформацію о замовленні. Відповідний оператор дивиться і починає виконувати його.

3. Оператор закінчив виконувати замовлення і повідомляє системі о готовності замовлення. Система фіксує зміну стану виробництва замовлення.

Розширення:

1а. Якщо система не може надати необхідну інформацію о замовленні:

1а1. Оператор повідомляє менеджера про проблему.

1а2. Менеджер вивчає ситуацію і надає оператору необхідну інформацію або вирішує проблему.

1а3. Оператор продовжує виконання замовлення згідно з основним сценарієм з кроку 2.

2а. Якщо під час виконання замовлення виникають технічні проблеми:

2а1. Оператор зупиняє роботу і оповіщає Менеджера о проблемі.

2а2. Менеджер оцінює проблему та визначає, чи може оператор вирішити її самостійно або потрібна зовнішня допомога.

2а3. Якщо оператор може вирішити проблему самостійно, він отримує інструкції та продовжує виконання замовлення.

2а4. Якщо потрібна зовнішня допомога, менеджер організовує необхідну підтримку.

2а5. Після вирішення проблеми оператор продовжує виконання замовлення згідно з основним сценарієм з кроку 2.

3 МОДЕЛЬ КОНЦЕПТУАЛЬНИХ КЛАСІВ

3.1 опис прецедентів з назвою можливих класів для реалізації кожного пункту сценарія.

Через аналіз двох най важливих варіантів використання «Замовлення послуги» і «Отримання послуги» ми отримаємо відповідні класи до кожного пункту їх сценарію з голови 2.2.1.1 та 2.2.1.2.

Таблиця 6.1 - Виявлення концептуальних класів на підставі ВВ «Замовлення послуги»

№ пункту	Зміст пункту основного сценарію	Можливий клас
1	Клієнт звертається за послугою. Касир створює нове замовлення у системі. Система фіксує касира, який наддає замовлення.	Order, Register, Operator
2	Касир запитує тип послуги. Клієнт говорить тип послуги. Касир вводить у систему сказаний тип послуги. Система підтверджує і фіксує.	Order, TypeList, TypeItem, Type, Register
3	Касир запитує кількість надаваної послуги. Клієнт говорить кількість потрібної послуги. Касир вводить у систему сказану кількість. Система підтверджує і фіксує.	Order, TypeItem, Register

4	Касир отримує з системи приблизний час виконання послуги. Касир говорить клієнту цей час.	Order, Term, Register
5	Касир запитує у системи вартість надаваної послуги. Касир говорить клієнту отриману з системи вартість послуги. Клієнт згоден з вартістю.	Order, PriceList, Money, Register
6	Клієнт дає відповідну кількість грошей, ПБ, контактний телефон. Касир вводить усе це у систему. Система перевіряє і фіксує замовлення, вибирає оператора, генерує квитанцію і змінює у виробничий стан замовлення.	Order, Register, PickUpStation, Operator, Ticket, Operator

Таблиця 6.2 - Виявлення концептуальних класів на підставі ВВ «Отримання послуги»

№ пункту	Зміст пункту основного сценарію	Можливий клас
1	Система повідомляє клієнта о кінцевим статусі послуги. Якщо замовлення готово, то повідомляє о його завершеності та потребує підійти до каси.	Order, Ticket, TicketList,
2	Клієнт повертається до касира та дає квитанцію. Касир вводить у систему номер квитанції. Система	Order, Ticket, PickUpStation

	перевіряє і Касир отримує з систему інформацію о статусі замовлення. Якщо замовлення готово, то касир підтверджує надання замовлення у системі та передає результат послуги клієнту.	
--	--	--

Таблиця 6.3 - Виявлення концептуальних класів на підставі ВВ «Виконання послуги»

№ пункту	Зміст пункту основного сценарію	Можливий клас
1	Система повідомляє о наявності невиконаного замовлення для конкретного оператора. Оператори дивяться і відповідний оператор запитує у системі більше додаткової інформації щодо замовлення.	Order, Ticket, TicketList, Operator
2	Система надає додаткову інформацію о замовленні. Відповідний оператор дивиться і починає виконувати його.	Order, Ticket, TicketList, Operator
3	Оператор закінчив виконувати замовлення і повідомляє системі о готовності замовлення. Система фіксує зміну стану виробництва замовлення.	Ticket, TicketList, Operator

3.2 діаграма концептуальних класів для першого варіанта використання

Через аналіз відношень класів у змісту пунктів першого прецеденту, з таблиці 6.1, була сформульована наступна діаграма на рисунку 2.1

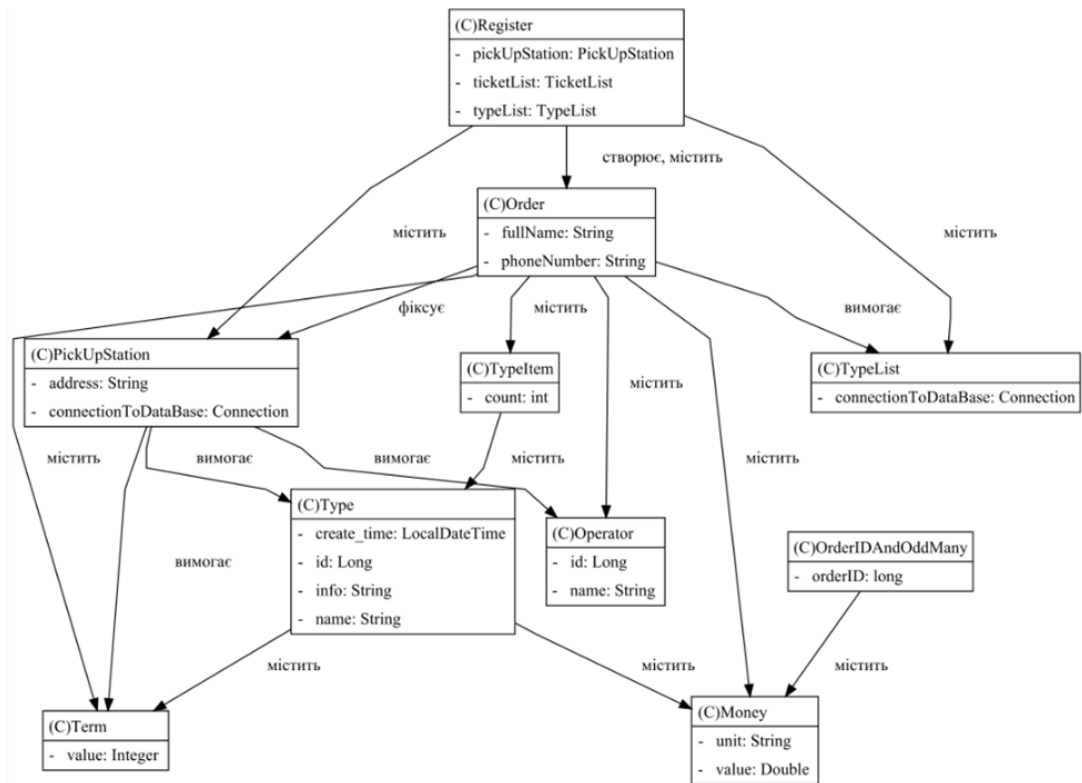


Рисунок 2.1 – Діаграма концептуальних класів прецеденту «Замовлення послуги»

3.3 діаграма концептуальних класів для першого і другого варіанта використання

Через аналіз відношень класів у змісті пунктів першого та другого прецеденту, з таблиці 6.1 і 6.2 відповідно, була сформульована наступна діаграма на рисунку 2.2

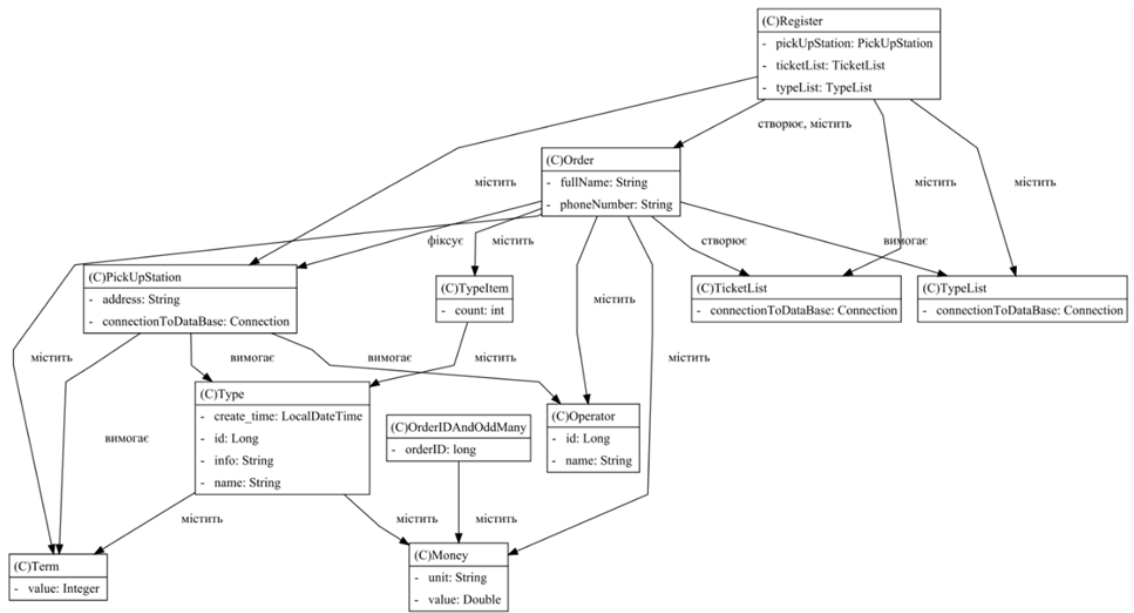


Рисунок 2.2 - Діаграма концептуальних класів прецеденту «Замовлення послуги» і «Отримання послуги», тобто загальна

3.4 загальна діаграма концептуальних класів:

Через аналіз відношень класів у змісті пунктів першого та другого прецеденту, з таблиці 6.1, 6.2 і 6.3 відповідно, була сформульована наступна діаграма на рисунку 2.3

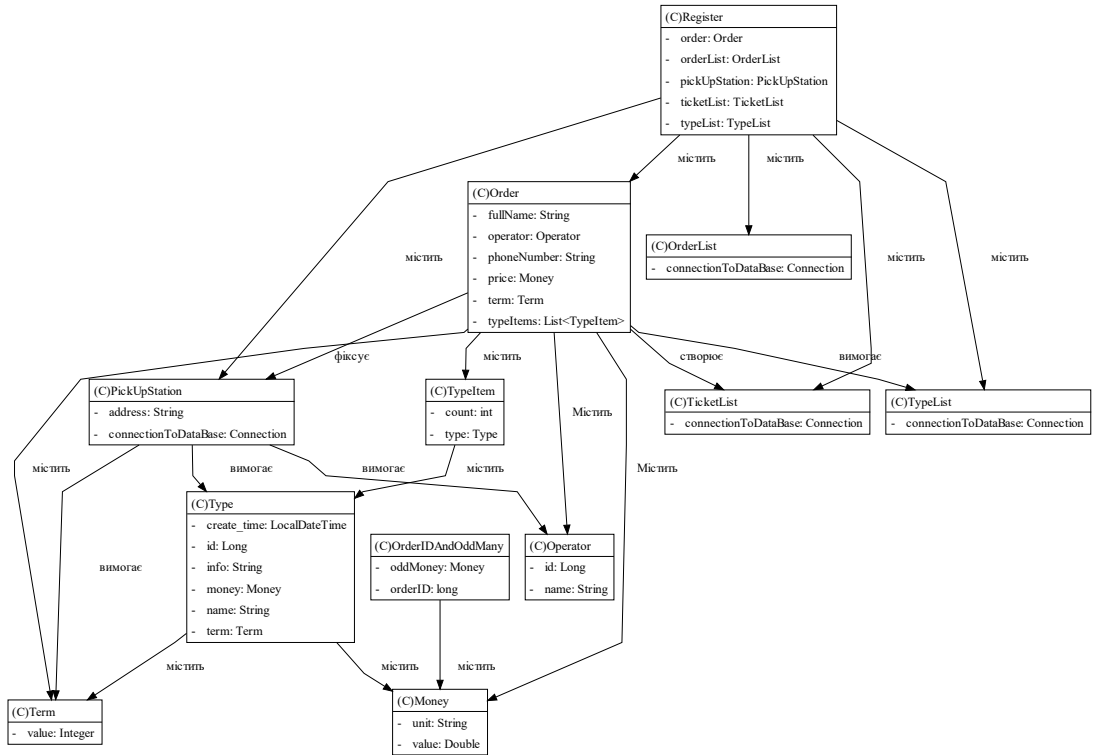


Рисунок 2.3 - Діаграма концептуальних класів прецеденту «Замовлення послуги», «Отримання послуги» і «Виконання послуги», тобто загальна

4 ДІАГРАМИ ВЗАЄМОДІЇ

4.1 діаграма системних операцій для «важливого» прецедентна:

Через аналіз пунктів прецеденту «Замовлення послуги» було отримано наступні системні операції з рисунку 3.1.

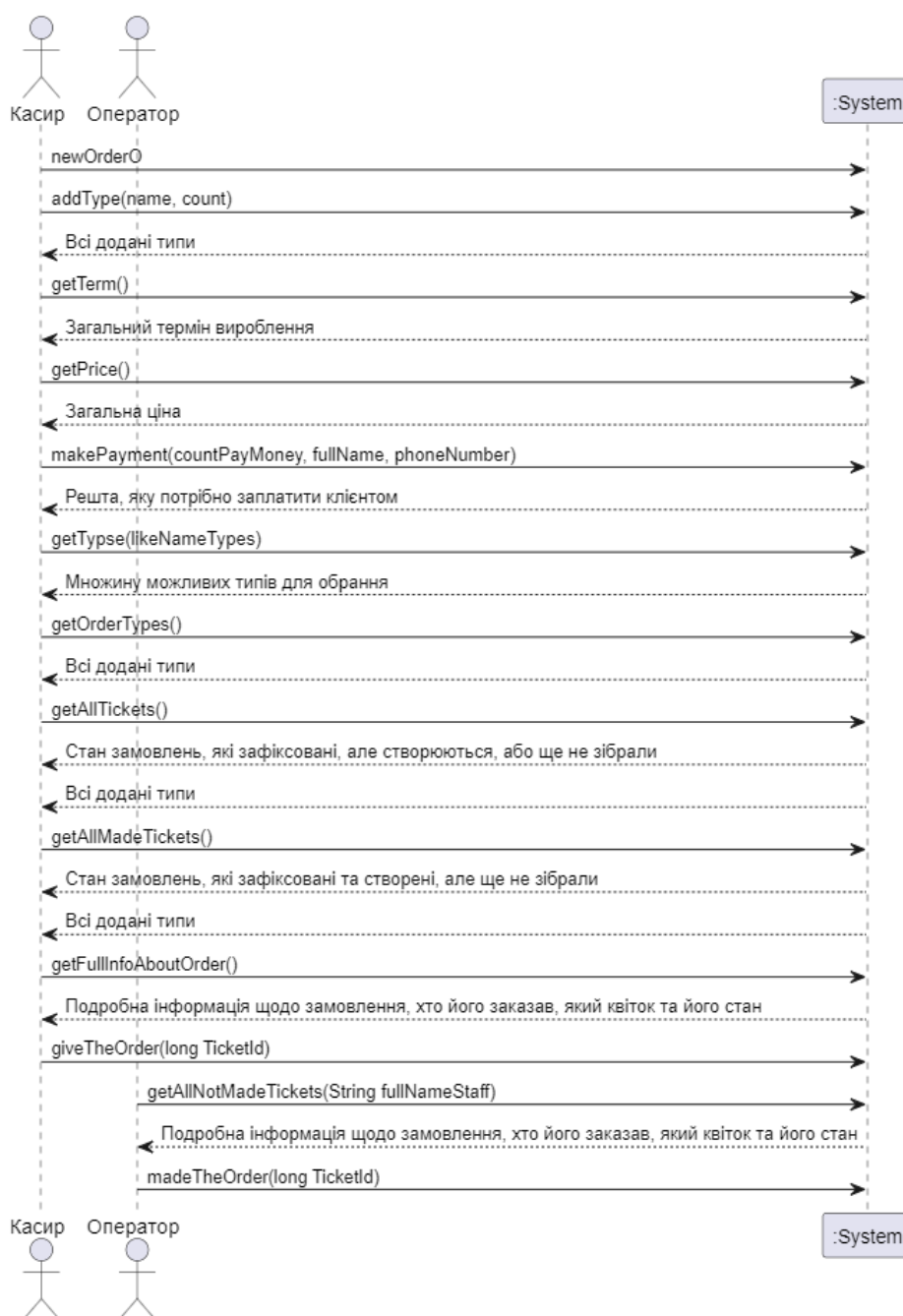


Рисунок 3.1 - Системні операції на підставі ВВ «Замовлення послуги»

4.2 обґрунтування вибору класу – контролер.

Клас Register підходить для реалізації шаблону проектування «Контролер», через взаємодію з ключовими елементами системи Order, Ticket та PickupStation.

4.3 діаграми взаємодії для кожного пункту сценарію «важливого» прецеденту:

Через аналіз відношень класів та самих змісту пунктів першого та другого прецеденту, були сформульовані наступні діаграма на рисунку 4.1, 4.2, 4.3, 4.4 та 4.5.

4.3.1 Проектне рішення newOrder

Клієнт звертається за послугою. Касир створює нове замовлення у системі. Система фіксує касира, який наддає замовлення.

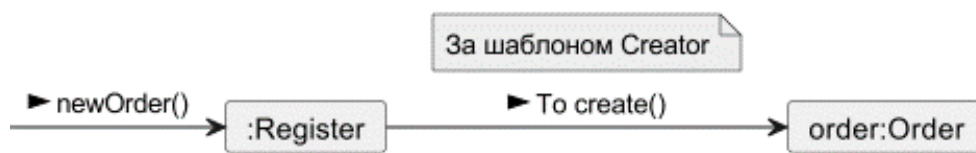


Рисунок 4.1 - Проектне рішення newOrder

4.3.2 Проектне рішення addType

Касир запитує тип послуги. Клієнт говорить тип послуги. Касир вводить у систему сказаний тип послуги. Система підтверджує і фіксує.

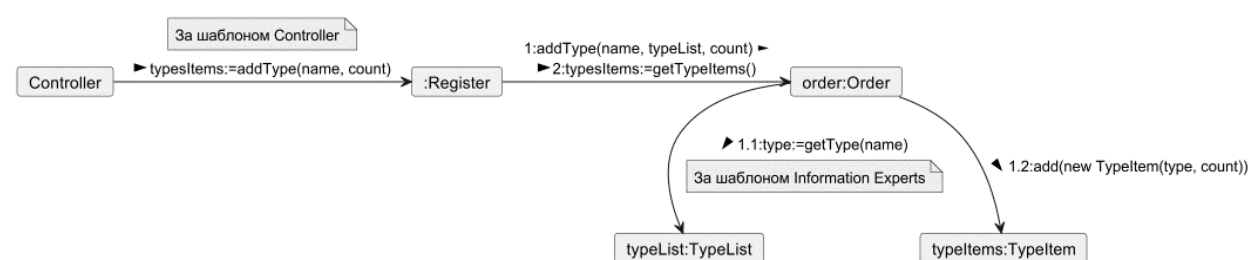


Рисунок 4.2 - Проектне рішення addType

4.3.3 Проектне рішення getTerm

Касир отримує з системи приблизний час виконання послуги. Касир говорить клієнту цей час.



Рисунок 4.3 - Проектне рішення getTerm

4.3.4 Проектне рішення getPrice

Касир отримує з системи приблизний час виконання послуги. Касир говорить клієнту цей час.



Рисунок 4.4 - Проектне рішення getPrice

4.3.5 Проектне рішення makePayment

Клієнт дає відповідну кількість грошей, ПІБ, контактний телефон. Касир вводить усе це у систему. Система перевіряє і фіксує замовлення, вибирає оператора, генерує квитанцію і змінює у виробничий стан замовлення.

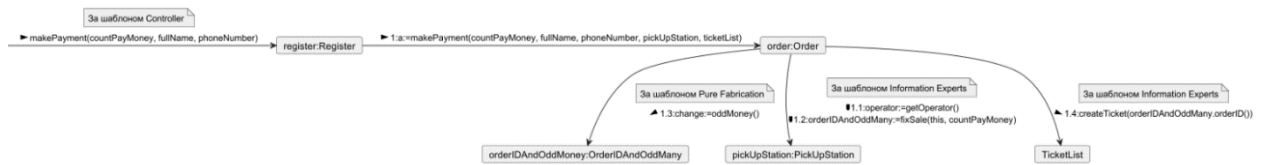


Рисунок 4.5 - Проектне рішення makePayment

4.4 обґрунтування прийнятого розподілу обов'язків з посиланням на шаблони проектування:

У першому був застосований шаблон проектування Create через потребу у створенні об'єкту для замовлення і взаємодії з ним за допомогою Register. У другому був застосований шаблон Controller для екземпляру з класом Register та information experts для екземпляру з типом TypeList, до яких проходять методи getType для додавання до екземпляру Order типу послуги. У третьому також використовується контролер. getTerm відповідає за розрахунок терміну. У четвертому рахується загальна ціна, також використовується контролер Register. Для п'ятого використовується шаблон information experts через PriceList та PickupStation для фіксування замовлення, отримання задачі за послугу та його розрахунок відповідно замовлення, OrderIDAndOddMany використовується за шаблоном Pure Fabrication для повертання даних з PickupStation fixSale(), також використовується Register з шаблоном конструктора для взаємодії з зовнішній системи, а саме ГКІ.

5 МОДЕЛЬ ДАНИХ

З приводу загального концептуального класу на рисунку 0.0 можливо отримати моделі для створення концептуальної моделі даних. Це можливо через заздалегідь спроектовану моделі орієнтованість та відокремлення бізнес-логіки від реалізації збереження даних.

5.1 Концептуальна модель даних

З рисунку 2.2 були взяті наступні класи: PriceList, TicketList, PickUpStation, TypeList, Ticket, Order, Operator. З цього можливо побудувати наступну концептуальну модель даних.

Рисунок 5.1 є зображенням концептуальної моделі даних для цих моделей та їх відношень.

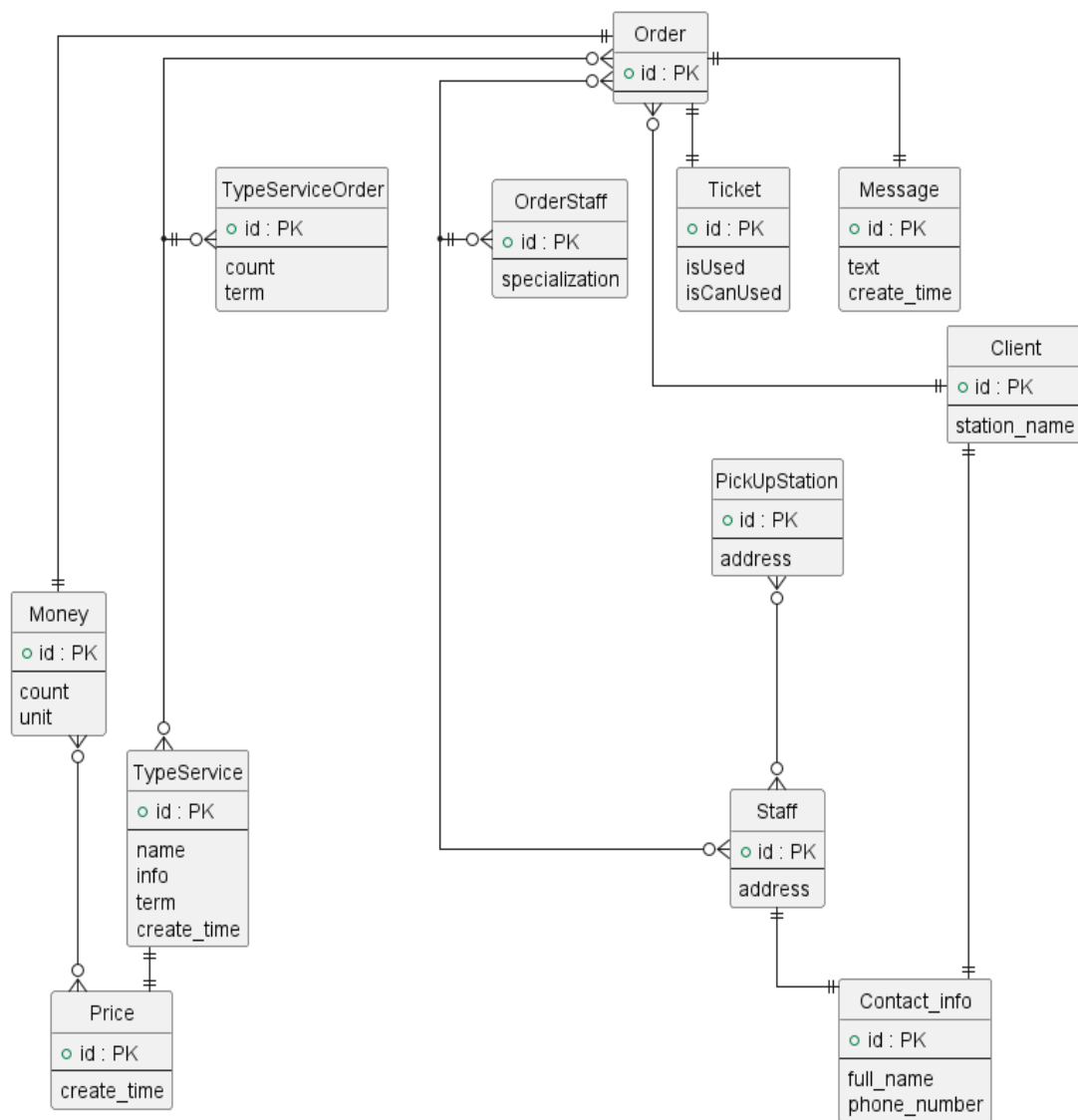


Рисунок 5.1: Концептуальні моделі даних для даного проекту

5.2 Реляційна модель даних

На рисунку 2 зображена реляційна модель даних для цих моделей та їх відношень.

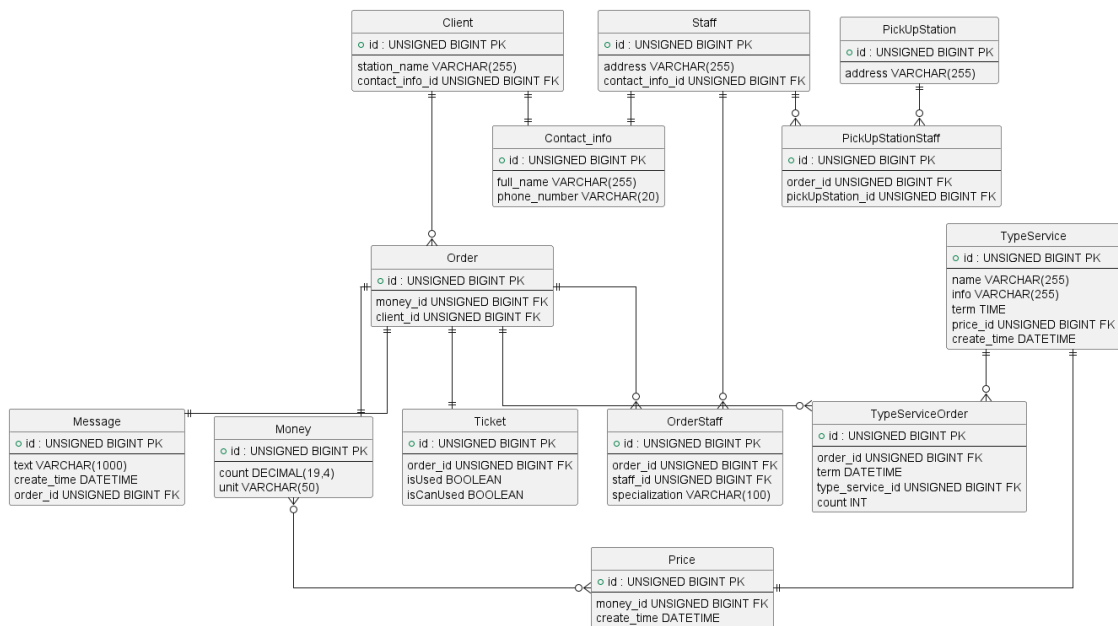


Рисунок 5.2. Реляційна модель даних для даного проекту

5.3 Обґрунтування вибору первинних ключів

У даному проектному рішенні бази даних було укладено, що у всіх таблиць буде первинним ключом окреме значення «id» для стандартизації та відсутності інших унікальних атрибутів.

6 ДІАГРАМИ ПРОГРАМНИХ КЛАСІВ

Через аналіз відношень класів, самих змісту пунктів першого та другого прецеденту, а також проектних рішень, були сформульовані наступні діаграма на рисунку 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 та 6.9.

6.1 Специфікація програмного класу Money

(C)Money
- unit: String - value: Double
+ Money(value: Double, unit: String) + sum(money: Money): Money + unit(): String + value(): Double

Рисунок 6.1 - Специфікація програмного класу Money

6.2 Специфікація програмного класу Operator

(C)Operator
- id: Long - name: String
+ Operator(id: Long, name: String) + id(): Long + name(): String

Рисунок 6.2 - Специфікація програмного класу Operator

6.3 Специфікація програмного класу Order

(C)Order
<ul style="list-style-type: none"> - fullName: String - operator: Operator - phoneNumber: String - price: Money - term: Term - typeItems: List<TypeItem>
<ul style="list-style-type: none"> - Order() + addType(name: String, typeList: TypeList, count: int) + create(): Order + getFullName(): String + getOperator(): Operator + getPhoneNumber(): String + getPrice(): Money + getTypeItems(): List<TypeItem> + makePayment(countPayMoney: Money, fullName: String, phoneNumber: String, pickupStation: PickupStation, ticketList: TicketList): Money + setFullName(fullName: String) + setOperator(operator: Operator) + setPhoneNumber(phoneNumber: String)

Рисунок 6.3 - Проектне рішення Order

6.4 Специфікація програмного класу OrderIDAndOddMoney

(C)OrderIDAndOddMoney
<ul style="list-style-type: none"> - oddMoney: Money - orderID: long
<ul style="list-style-type: none"> + OrderIDAndOddMany(orderID: long, oddMoney: Money) + oddMoney(): Money + orderID(): long

Рисунок 6.4 - Проектне рішення OrderIDAndOddMoney

6.5 Специфікація програмного класу PickupStation

(C)PickUpStation
<ul style="list-style-type: none"> - address: String - connectionToDataBase: Connection
<ul style="list-style-type: none"> + PickUpStation(connectionToDataBase: Connection, address: String) + address(): String - checkAffectRows(affectedRows: int) + connectionToDataBase(): Connection + fixSale(order: Order, countPayMoney: Money): OrderIDAndOddMany - getId(statement: PreparedStatement): long + getOperator(): Operator - insertClient(id: long): long - insertContactInfo(order: Order): long - insertMoney(order: Order): long - insertOrder(idMoney: long, idClient: long): long - insertOrderStaff(idOrder: long, operator: Operator) - insertTypeServiceOrder(idOrder: long, type: TypeItem, term: Term)

Рисунок 6.5 - Специфікація програмного класу TypeItem

6.6 Специфікація програмного класу Register

(C)Register
<ul style="list-style-type: none"> - order: Order - orderList: OrderList - pickUpStation: PickUpStation - ticketList: TicketList - typeList: TypeList
<ul style="list-style-type: none"> + Register(typeList: TypeList, pickUpStation: PickUpStation, ticketList: TicketList, orderList: OrderList) + addType(name: String, count: int): List<TypeItem> + getAllMadeTickets(): List<Ticket> + getAllNotMadeTickets(fullNameStaff: String): List<Ticket> + getAllTickets(): List<Ticket> + getFullInfoAboutOrder(ticketId: long): OrderViewAndTypesViews + getOrderTypes(): List<TypeItem> + getPrice(): Money + getTerm(): Term + getTypes(likeNameTypes: String): List<Type> + giveTheOrder(ticketId: long) + madeTheOrder(ticketId: long) + makePayment(countPayMoney: Money, fullName: String, phoneNumber: String): Money + newOrder()

Рисунок 6.6 - Специфікація програмного класу Register

6.7 Специфікація програмного класу Term

(C)Term
- value: Integer
+ Term(value: Integer)
+ sum(term: Term): Term
+ toLocalTime(): LocalTime
+ value(): Integer

Рисунок 6.7 - Специфікація програмного класу Term

6.8 Специфікація програмного класу Ticket

(C)Ticket
- fullName: String - id: long - isDone: boolean - isReady: boolean - name: String - orderId: long - phoneNumber: String
+ Ticket(id: long, orderId: long, name: String, isReady: boolean, isDone: boolean, fullName: String, phoneNumber: String) + getFullName(): String + getId(): long + getIsDone(): boolean + getIsReady(): boolean + getName(): String + getOrderId(): long + getPhoneNumber(): String + setFullName(fullName: String) + setId(id: long) + setIsDone(done: boolean) + setIsReady(ready: boolean) + setName(name: String) + setOrderId(orderId: long) + setPhoneNumber(phoneNumber: String)

Рисунок 6.8 - Специфікація програмного класу Ticket

6.9 Специфікація програмного класу TicketList

(C)TicketList
- connectionToDataBase: Connection + TicketList(connectionToDataBase: Connection) - checkAffectRows(affectedRows: int) + connectionToDataBase(): Connection + createTicket(idOrder: long): long + getAllMadeTickets(): List<Ticket> + getAllNotMadeTickets(fullNameStaff: String): List<Ticket> + getAllTickets(): List<Ticket> - getId(statement: PreparedStatement): long - getTickets(statement: PreparedStatement): List<Ticket> - insertTicket(idOrder: long): long + setEndOrderTicket(id: long) + setMadeOrderTicket(id: long)

Рисунок 6.9 - Специфікація програмного класу TicketList

6.10 Специфікація програмного класу Type

(C)Type
- create_time: LocalDateTime - id: Long - info: String - money: Money - name: String - term: Term
+ Type(id: Long, name: String, info: String, term: Term, money: Money, create_time: LocalDateTime) + create_time(): LocalDateTime + id(): Long + info(): String + money(): Money + name(): String + term(): Term

Рисунок 6.10 - Специфікація програмного класу Type

6.11 Специфікація програмного класу **TypeItem**

(C)TypeItem
<ul style="list-style-type: none"> - count: int - type: Type
<ul style="list-style-type: none"> + TypeItem(type: Type, count: int) + count(): int + getMoney(): Money + getTerm(): Term + type(): Type

Рисунок 6.11 - Специфікація програмного класу **TypeItem**

6.12 Специфікація програмного класу **TypeList**

(C)TypeList
<ul style="list-style-type: none"> - connectionToDataBase: Connection
<ul style="list-style-type: none"> + TypeList(connectionToDataBase: Connection) + getType(typeName: String): Type + getTypes(likeNameTypes: String): List<Type>

Рисунок 6.12 - Специфікація програмного класу **TypeList**

6.12 Специфікація програмного класу **OrderList**

(C)OrderList
- connectionToDataBase: Connection
+ OrderList(connectionToDataBase: Connection)
+ connectionToDataBase(): Connection
+ getFullInfoAboutOrder(ticketId: long): OrderViewAndTypesViews

Рисунок 6.13 - Специфікація програмного класу OrderList

6.12 Специфікація програмного класу OrderViewAndTypesViews

(C)OrderViewAndTypesViews
- orderView: OrderView
- typeViews: List<TypeView>
+ OrderViewAndTypesViews(orderView: OrderView, typeViews: List<TypeView>)
+ orderView(): OrderView
+ typeViews(): List<TypeView>

Рисунок 6.14 - Специфікація програмного класу OrderViewAndTypesViews

6.13 обґрунтування прийнятих рішень, щодо визначення методів і атрибутів класів; Діаграма програмних класів.

Для класу Register методи впливають із системних операцій, діаграм взаємодій та необхідності у створенні екземпляра класу. Атрибути так само з діаграм взаємодій. Цей клас є точкою входу даного модуля та виконання прецеденту.

Для класу Order методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу PickupStation методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Загалом способи представляють функції фіксації та доступу до вільного Оператора.

Для класу `TypeList` методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Функціонал методів являє собою видачу самого типу або часу виготовлення даного типу.

Для класу `Term` методи представлятимуться так само з діаграм взаємодій. Метод статичний необхідний створення екземпляра класу, атрибут необхідний зберігання асоційованих даних. Функція способу надання тривалості.

Для класу `Type` методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `Money` методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `Operator` методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `OrderIDAndOddMoney` методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `Ticket` методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `TicketList` методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу `OrderList` методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

6.14 обґрунтування виявлених відношень між класами:

Для класу Register методи впливають із системних операцій, діаграм взаємодій та необхідності у створенні екземпляра класу. Атрибути так само з діаграм взаємодій. Цей клас є точкою входу даного модуля та виконання прецеденту.

Для класу Order методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу PickUpStation методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Загалом способи представляють функції фіксації та доступу до вільного Оператора.

Для класу TypeList методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Функціонал методів являє собою видачу самого типу або часу виготовленняданого типу.

Для класу Term методи представлятимуться так само з діаграм взаємодій. Метод статичний необхідний створення екземпляра класу, атрибут необхідний зберігання асоційованих даних. Функція способу надання тривалості.

Для класу Ticket методи представлятимуться так само з діаграм взаємодій. Метод статичний необхідний створення екземпляра класу, атрибут необхідний зберігання асоційованих даних. Функція способу надання тривалості.

Для класу Type методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу Money методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу Operator методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу OrderIDAndOddMoney методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу Ticket методи з атрибутами впливають із діаграм взаємодій, а типи з пункту варіанти використання прецеденту. Методи доступу необхідні подальшого використання інших частинах системи, які з даного модуля.

Для класу TicketList методи представлятимуться так само з діаграм взаємодій. Метод конструктор необхідний створення екземпляра класу, атрибут необхідний визначення шляху бази даних. Функціонал методів являє собою видачу самого типу або часу виготовленняданого типу.

7 ПРОГРАМНІ КЛАСИ

З приводу минулого проектування слідує наступний код модуля, який приведено у додатку А. У додатку Б знаходиться код контролерів для інтерфейсу, точка входу, XML розмітка інтерфейсу, автоматичні тести та module-info. У додатку В знаходиться структура папок модуля. У додатку Г приведений код конфігурації для Maven. У додатку Д записана інструкція користувача.

8 ЗАПИТИ ДО БАЗИ ДАНИХ

Код створення бази даних та додавання таблиць наступний:

```
CREATE DATABASE IF NOT EXISTS `PhotocopyPoint`;
```

```
USE `PhotocopyPoint`;
```

```
CREATE TABLE IF NOT EXISTS `Contact_info` (  
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    `full_name` VARCHAR(255) NOT NULL,  
    `phone_number` VARCHAR(14) NOT NULL,  
    PRIMARY KEY (`id`)  
);
```

```
CREATE TABLE IF NOT EXISTS `PickUpStation` (  
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    `address` VARCHAR(255) NOT NULL,  
    PRIMARY KEY (`id`)  
);
```

```
CREATE TABLE IF NOT EXISTS `Money` (  
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    `count` DECIMAL(19,4) NOT NULL,  
    `unit` VARCHAR(50) NOT NULL,  
    PRIMARY KEY (`id`)  
);
```

```
CREATE TABLE IF NOT EXISTS `Price` (  
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    `money_id` BIGINT UNSIGNED NOT NULL,
```

```

`create_time` DATETIME NOT NULL,
PRIMARY KEY (`id`),
FOREIGN KEY (`money_id`) REFERENCES `Money`(`id`)
);

```

```

CREATE TABLE IF NOT EXISTS `TypeService` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `info` VARCHAR(255) NOT NULL,
  `term` TIME NOT NULL,
  `price_id` BIGINT UNSIGNED NOT NULL,
  `create_time` DATETIME NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`price_id`) REFERENCES `Price`(`id`)
);

```

```

CREATE TABLE IF NOT EXISTS `Client` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `station_name` VARCHAR(255) NOT NULL,
  `contact_info_id` BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`contact_info_id`) REFERENCES `Contact_info`(`id`)
);

```

```

CREATE TABLE IF NOT EXISTS `Staff` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `address` VARCHAR(255) NOT NULL,
  `contact_info_id` BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`contact_info_id`) REFERENCES `Contact_info`(`id`)
);

```

);

```
CREATE TABLE IF NOT EXISTS `Order` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `money_id` BIGINT UNSIGNED NOT NULL,
  `client_id` BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`money_id`) REFERENCES `Money`(`id`),
  FOREIGN KEY (`client_id`) REFERENCES `Client`(`id`)
);
```

```
CREATE TABLE IF NOT EXISTS `Ticket` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `order_id` BIGINT UNSIGNED NOT NULL,
  `isUsed` BOOLEAN NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`order_id`) REFERENCES `Order`(`id`)
);
```

```
CREATE TABLE IF NOT EXISTS `Message` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `text` VARCHAR(1000) NOT NULL,
  `create_time` DATETIME NOT NULL,
  `order_id` BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`order_id`) REFERENCES `Order`(`id`)
);
```

```
CREATE TABLE IF NOT EXISTS `TypeServiceOrder` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
```



```

`order_id` BIGINT UNSIGNED NOT NULL,
`term` DATETIME NOT NULL,
`type_service_id` BIGINT UNSIGNED NOT NULL,
`count` INT NOT NULL,
PRIMARY KEY (`id`),
FOREIGN KEY (`order_id`) REFERENCES `Order`(`id`),
FOREIGN KEY (`type_service_id`) REFERENCES `TypeService`(`id`)
);

```

```

CREATE TABLE IF NOT EXISTS `OrderStaff` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `order_id` BIGINT UNSIGNED NOT NULL,
  `staff_id` BIGINT UNSIGNED NOT NULL,
  `specialization` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`order_id`) REFERENCES `Order`(`id`),
  FOREIGN KEY (`staff_id`) REFERENCES `Staff`(`id`)
);

```

```

CREATE TABLE IF NOT EXISTS `PickUpStationStaff` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `staff_id` BIGINT UNSIGNED,
  `pickUpStation_id` BIGINT UNSIGNED,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`staff_id`) REFERENCES `Staff`(`id`),
  FOREIGN KEY (`pickUpStation_id`) REFERENCES `PickUpStation`(`id`)
);

```

Додавання даних.

```
INSERT INTO `Contact_info` (`full_name`, `phone_number`) VALUES
('Іван Іванов', '+380501234567'),
('Петро Петренко', '+380671234568');
```

```
INSERT INTO `PickUpStation` (`address`) VALUES
('Київ, вул. Хрещатик, 1'),
('Львів, просп. Свободи, 45');
```

```
INSERT INTO `Money` (`count`, `unit`) VALUES
(100.00, 'UAH'),
(200.00, 'UAH');
```

```
INSERT INTO `Price` (`money_id`, `create_time`) VALUES
(1, NOW()),
(2, NOW());
```

```
INSERT INTO `TypeService` (`name`, `info`, `term`, `price_id`, `create_time`)
VALUES
('Друк', 'Чорно-білий друк', '00:01:00', 1, NOW()),
('Копіювання', 'Кольорове копіювання', '00:02:30', 2, NOW());
```

```
INSERT INTO `Client` (`station_name`, `contact_info_id`) VALUES
('Київська станція', 1);
```

```
INSERT INTO `Staff` (`address`, `contact_info_id`) VALUES
('Київ, вул. Саксаганського, 12', 2);
```

```
INSERT INTO `Order` (`money_id`, `client_id`) VALUES
```

```
(1, 1);
```

```
INSERT INTO `Ticket` (`order_id`, `isUsed`) VALUES
(1, FALSE);
```

```
INSERT INTO `Message` (`text`, `create_time`, `order_id`) VALUES
('Повідомлення для оператора принтера: додаткова інформація', NOW(), 1);
```

```
INSERT INTO `TypeServiceOrder` (`term`, `order_id`, `type_service_id`, `count`)
VALUES
(NOW() + INTERVAL 1 MINUTE, 1, 1, 10);
INSERT INTO `OrderStaff` (`staff_id`, `order_id`, `specialization`) VALUES
(1, 1, 'Оператор принтера');
```

```
INSERT INTO `PickUpStationStaff` (`staff_id`, `pickUpStation_id`) VALUES
(1, 1);
```

Отримати усі типи послуг та їх ціну за новою редакцією та назвою яка починається на «Д»

```
SELECT TS.name,
       TS.id,
       TS.info,
       TS.term,
       PM.count,
       PM.unit,
       TS.create_time
FROM PhotocopyPoint.TypeService TS
INNER JOIN (
  SELECT name, MAX(create_time) AS max_create_time
  FROM PhotocopyPoint.TypeService
```

```

        GROUP BY name
    ) AS latest_services
    ON    TS.name    =    latest_services.name    AND    TS.create_time    =
latest_services.max_create_time

    INNER JOIN PhotocopyPoint.Price PP ON TS.price_id = PP.id
    INNER JOIN PhotocopyPoint.Money PM ON PP.money_id = PM.id
    WHERE TS.name LIKE CONCAT('%', "д", '%')
    ORDER BY TS.id DESC;

```

Отримати тип послуги та його ціну за назвою яка починається на «д»

```

SELECT TS.id,
       TS.name,
       TS.info,
       TS.term,
       PM.count,
       PM.unit,
       TS.create_time
From PhotocopyPoint.Money PM
inner join PhotocopyPoint.Price PP on PM.id = PP.money_id
inner join PhotocopyPoint.TypeService TS on PP.id = TS.price_id
where TS.name like CONCAT('%', "д", '%')
ORDER BY TS.create_time DESC LIMIT 1;

```

Отримати усі статуси створення замовлення.

```

SELECT PT.id,
       PT.order_id,
       GROUP_CONCAT(TS.name SEPARATOR ', ') AS all_names,
       PT.isUsed,
       Ci.full_name,
       Ci.phone_number

```

FROM PhotocopyPoint.Ticket PT

INNER JOIN PhotocopyPoint.`Order` PO ON PO.id = PT.order_id

INNER JOIN PhotocopyPoint.Client C ON PO.client_id = C.id

INNER JOIN PhotocopyPoint.Contact_info Ci ON C.contact_info_id = Ci.id

INNER JOIN PhotocopyPoint.TypeServiceOrder TSO ON PO.id =
TSO.order_id

INNER JOIN PhotocopyPoint.TypeService TS ON TSO.type_service_id =
TS.id

GROUP BY PT.id, PT.order_id, PT.isUsed, Ci.full_name, Ci.phone_number

ORDER BY PT.id ASC;

Отримання ціни типу з назвою «Копіювання»:

SELECT PhotocopyPoint.Money.count, PhotocopyPoint.Money.unit,
PhotocopyPoint.Price.create_time

From PhotocopyPoint.Money

inner join PhotocopyPoint.Price on PhotocopyPoint.Price.money_id =
PhotocopyPoint.Money.id

inner join PhotocopyPoint.TypeService on PhotocopyPoint.TypeService.price_id =
PhotocopyPoint.Price.id

where PhotocopyPoint.TypeService.name = "Копіювання"

ORDER BY PhotocopyPoint.Price.create_time DESC LIMIT 1;

Отримання оператора принтера на станції за адресою «Київ, вул. Хрещатик,
1», який зараз не зайнятий:

SELECT PhotocopyPoint.Contact_info.id, PhotocopyPoint.Contact_info.full_name
From PhotocopyPoint.Contact_info

inner join PhotocopyPoint.Staff on PhotocopyPoint.Staff.contact_info_id =
PhotocopyPoint.Contact_info.id

inner join PhotocopyPoint.PickUpStationStaff on
PhotocopyPoint.PickUpStationStaff.staff_id = PhotocopyPoint.Staff.id

```

left join PhotocopyPoint.OrderStaff on PhotocopyPoint.OrderStaff.staff_id =
ALL(Select PhotocopyPoint.Staff.id from PhotocopyPoint.Staff inner join
PhotocopyPoint.PickUpStation on PhotocopyPoint.PickUpStation.id =
PhotocopyPoint.PickUpStationStaff.pickUpStation_id WHERE
PhotocopyPoint.PickUpStation.address = "missingValue" )
left join PhotocopyPoint.`Order` on PhotocopyPoint.`Order`.id =
PhotocopyPoint.OrderStaff.order_id
left join PhotocopyPoint.TypeServiceOrder on
PhotocopyPoint.TypeServiceOrder.id = PhotocopyPoint.`Order`.`id`
where (PhotocopyPoint.TypeServiceOrder.term < NOW() or
(PhotocopyPoint.TypeServiceOrder.term is null)) LIMIT 1;

```

Отримати час створення замовлення під назвою «Копіювання»:

```

SELECT PhotocopyPoint.TypeService.term,
PhotocopyPoint.TypeService.create_time
From PhotocopyPoint.TypeService
where TypeService.name = "Копіювання"
ORDER BY PhotocopyPoint.TypeService.create_time DESC LIMIT 1;

```

Отримання додаткової інформації щодо типу замовлення під назвою «Копіювання»:

```

SELECT PhotocopyPoint.TypeService.name, PhotocopyPoint.TypeService.info,
PhotocopyPoint.TypeService.create_time From PhotocopyPoint.TypeService
where PhotocopyPoint.TypeService.name = "Копіювання"
ORDER BY PhotocopyPoint.TypeService.create_time DESC LIMIT 1;

```

Отримати усіх клієнтів, які мають по батькові Іванов:

```

Select Contact_info.full_name, Contact_info.phone_number, `Client`.station_name
from `Client`
inner join Contact_info on `Client`.contact_info_id = Contact_info.id
where Contact_info.full_name like '%Іванов%';

```

Отримати замовлення, у який термін закінчення створення в заданий період:

```
SELECT * FROM `TypeServiceOrder` where TypeServiceOrder.term BETWEEN
'2024-05-27 08:00:0' and '2024-05-27 18:00:00';
```

Скільки створено замовлень за ім'ям даного клієнта:

```
SELECT count(*) FROM `Order` inner join Client on Client.id = `Order`.id INNER
JOIN Contact_info on Client.contact_info_id = Contact_info.id WHERE
Contact_info.full_name LIKE '%Іван%';
```

Скільки кожний робітник виконав замовлень за весь час:

```
SELECT Contact_info.full_name, COUNT(OrderStaff.id) From Contact_info
inner join Staff on Staff.contact_info_id = Contact_info.id
inner join PickupStationStaff on PickupStationStaff.staff_id = Staff.id
left join OrderStaff on Staff.id = OrderStaff.staff_id
left join `Order` on `Order`.id = OrderStaff.order_id
left join TypeServiceOrder on TypeServiceOrder.id = `Order`.`id`
where TypeServiceOrder.term < NOW() or (TypeServiceOrder.term is null)
GROUP BY Contact_info.full_name ORDER BY Contact_info.full_name;
```

Отримати терміни виготовлення замовлення для типу замовлення «Друк»:

```
SELECT TypeServiceOrder.term from TypeServiceOrder where
TypeServiceOrder.type_service_id = ALL(SELECT TypeService.`id` from TypeService
WHERE TypeService.name = 'Друк');
```

Отримати всі ім'я та, якщо є, суму потрачену на замовлення:

```
SELECT Contact_info.full_name, (
SELECT SUM(Money.count)
FROM Money
```

```

inner join `Order` on `Order`.`money_id` = Money.id
inner join `Client` on `Order`.client_id = `Client`.id
WHERE `Client`.`contact_info_id` = Contact_info.id
) AS total_money
FROM Contact_info;

```

Хто з робітників не повинен виконувати замовлення у цьому місяці:

```

SELECT Contact_info.full_name
FROM Contact_info
INNER JOIN Staff on Staff.contact_info_id = Contact_info.id

```

LEFT JOIN

```

(select staff_id from OrderStaff
inner JOIN `Order` on `Order`.`id` = OrderStaff.order_id
inner JOIN TypeServiceOrder on TypeServiceOrder.order_id = `Order`.`id`
where (TypeServiceOrder.term BETWEEN DATE_FORMAT(NOW(), "%y-%m-01") and LAST_DAY(NOW())) as
on ss.staff_id = Staff.id
where ss.staff_id is null;

```

SS

Отримати усі ціни типи послуг за деякою додатковою інформацією окрім перерахованих:

```

SELECT TypeService.name, Money.`count`, Money.unit from Money
INNER JOIN Price on Price.money_id = Money.id
INNER JOIN TypeService on TypeService.price_id = Price.id
WHERE TypeService.info not in ("Чорно-білий друк")
GROUP BY TypeService.name, Money.`count`, Money.unit ORDER BY
TypeService.name;

```


Хто з робітників ні разу не отримував завдання робити замовлення:

```
SELECT Contact_info.full_name FROM Contact_info
INNER JOIN Staff on Staff.contact_info_id = Contact_info.id
where not EXISTS (SELECT OrderStaff.order_id FROM OrderStaff
WHERE OrderStaff.staff_id = Staff.id);
```

Хто з робітників отримав максимальну кількість замовлення і хто не отримав жодного за весь час:

```
SELECT name, MAX(count_staff), 'Хто з робітників отримав максимальну
кількість замовлення за весь час' FROM (SELECT Contact_info.full_name as name,
COUNT(OrderStaff.id) as count_staff From Contact_info
inner join Staff on Staff.contact_info_id = Contact_info.id
inner join PickupStationStaff on PickupStationStaff.staff_id = Staff.id
left join OrderStaff on Staff.id = OrderStaff.staff_id
left join `Order` on `Order`.id = OrderStaff.order_id
left join TypeServiceOrder on TypeServiceOrder.id = `Order`.`id`
where TypeServiceOrder.term < NOW()
GROUP BY Contact_info.full_name ORDER BY Contact_info.full_name) as t
GROUP BY name
UNION
SELECT name, count_staff, 'хто не отримав жодного за весь час'
from (SELECT Contact_info.full_name as name, 0 as count_staff FROM
Contact_info
INNER JOIN Staff on Staff.contact_info_id = Contact_info.id
where not EXISTS (SELECT OrderStaff.order_id FROM OrderStaff
WHERE OrderStaff.staff_id = Staff.id)) as l
GROUP BY name ORDER BY name;
```

9 ТЕСТУВАННЯ

Таблиця 7.1 Тест-кейси для методу getTypes()

Т ест-кейс	Опис	Вхідні дані	Очікуваний результат
Т C1	Пере вірка отримання списку типів за частковим збігом	"д"	Список з одним елементом: Type(5L, "Друк", "Чорно-білий друк", новий Term(60), новий Money(100.0000, "UAH"), Mysql.dbDateTimeToLocalDateT ime("2024-05-27 11:48:27"))
Т C2	Виня ток при відсутності типу	"незвича йний тип"	NotExistTypeException

Таблиця 7.2 Тест-кейси для методу addType()

Т ест-кейс	Опис	Вхідні дані	Очікуваний результат
Т C1	Пере вірка додавання нового типу	"Копіюв ання"	Type(2L, "Копіювання", "Кольорове копіювання", новий Term(150), новий Money(200.0000, "UAH"), Mysql.dbDateTimeToLocalDateT ime("2024-05-27 11:48:24"))

Таблиця 7.3 Тест-кейси для методу getAllTickets()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка отримання всіх квитків	Немає	Список квитків з одним елементом: Ticket(1, 1, "Друк", false, "Іван Іванов", "+380501234567")
ТС2	Перевірка відсутності квитків	Порожня база даних	Порожній список квитків
ТС3	Перевірка наявності кількох квитків	Декілька квитків у базі даних	Список квитків відповідно до записів у базі даних

Таблиця 7.4 Тест-кейси для методу createTicket()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Створення нового квитка для існуючого замовлення	idOrder = 1 (існуюче замовлення)	Новий квиток створено, idTicket збігається з записом у базі даних
ТС2	Створення нового квитка для неіснуючого замовлення	idOrder = 9999 (неіснуюче замовлення)	Виняток або відповідна помилка при створенні квитка
ТС3	Перевірка створення кількох квитків	idOrder = 1 (існуюче замовлення),	Створення кількох квитків з унікальними idTicket

		кілька викликів методу	для одного замовлення
--	--	---------------------------	--------------------------

Таблиця 7.5 Тест-кейси для методу getTerm()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка обчислення терміну для доданого типу	"Друк", 57 хвилин	Термін 00:57:00
ТС2	Перевірка обчислення терміну для іншого типу	"Копіювання", 30 хвилин	Термін 00:07:30
ТС3	Перевірка обчислення терміну для декількох типів	"Друк", 30 хвилин і "Копіювання", 20 хвилин	Термін 00:50:00

Таблиця 7.6 Тест-кейси для методу getPrice()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка обчислення ціни для доданого типу	"Друк", 57 одиниць	Ціна 5700.0000 UAH
ТС2	Перевірка обчислення ціни для іншого типу	"Копіювання", 30 одиниць	Ціна 7500.0000 UAH

ТС3	Перевірка обчислення ціни для декількох типів	"Друк", одиниць "Копіювання", одиниць	30 і 20	Ціна 3000.0000 UAH
-----	--	--	---------------	-----------------------

Таблиця 7.7 Тест-кейси для методу makePayment()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка розрахунку здачі при оплаті	Платіж 6000 UAH за "Друк", 57 одиниць	Здача 300 UAH
ТС2	Перевірка розрахунку здачі при точній оплаті	Платіж 5700 UAH за "Друк", 57 одиниць	Здача 0 UAH
ТС3	Перевірка обробки недостатньої оплати	Платіж 5000 UAH за "Друк", 57 одиниць	Виняток або відповідна помилка

Таблиця 7.8 Тест-кейси для методу getOperator()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка отримання оператора для конкретної станції	Станція "Київ, вул. Хрещатик, 1"	Оператор (2L, "Петро Петренко")

TC2	Перевірка відсутності оператора для неіснуючої станції	Станція "Неіснуюча адреса"	Виняток або null
-----	---	----------------------------------	---------------------

Таблиця 7.9 Тест-кейси для методу fixSale()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC1	Перевірка завершення продажу та розрахунку здачі	Замовлення з типом "Друк", кількість 3, оплата 500 UAN	Здача 200 - (ціна замовлення), запис у базі даних з правильним ім'ям та ідентифікатором
TC2	Перевірка завершення продажу з точним платежем	Замовлення з типом "Друк", кількість 3, оплата 300	Здача 0, запис у базі даних з правильним ім'ям та ідентифікатором
TC3	Перевірка завершення продажу з недостатнім платежем	Замовлення з типом "Друк", кількість 3, оплата 150	Виняток або відповідна помилка

ВИСНОВКИ

У даній курсовій роботі були поглиблені та закріплені знання, одержаних при вивченні дисципліни «Конструювання програмного забезпечення», «Бази даних», та набуття практичних навичок у проектуванні та налагодженні програм, що застосовують класи та об'єкти.

Було розроблене програма з графічним користувацьким інтерфейсом, яка працює зі списками об'єктів. Інформація для списків зберігається у базі даних та беруться через запити до неї.

Були створені сутності у базі даних «Оператор», «Клієнт», «Послуга» з відповідними та потрібними, через вимоги, властивостями.

Реалізована композиція та успадкування. У роботі описано та реалізовано інтерфейс користувача.

Було реалізовано та протестовано створення замовлення, ведення користувачем даних за деякими правилами, фіксування у базу даних та отримання стану замовлення.

СПИСОК ЛІТЕРАТУРИ

1 Кунгурцев О.Б. Основи програмування на мові Java. Середовище Net Beans./О.Б. Кунгурцев – Одеса: ВМВ, 2006. – 182 с

2 Методичні вказівки до курсової роботи з дисципліни «Об’єктно – орієнтоване програмування» для студентів першого рівня вищої освіти (бакалавр). Галузь знань - 12 Інформаційні технології. Спеціальність 121 Інженерія програмного забезпечення / Укл.: О. Б. Кунгурцев. – Одеса: «Одеська політехніка», 2022. –24 с.

3 “Руководство по языку программирования Java” // metanit.com [2023]. Дата відновлення: 20.09.2023. URL: <https://metanit.com/java/tutorial/> (дата звернення: 05.12.2023).

4 “Руководство по JavaFX” // metanit.com [2023]. Дата відновлення: 23.09.2021. URL: <https://metanit.com/java/javafx/> (дата звернення: 05.12.2023).

ДОДАТОК А КОД МОУДЛЯ

Money.java

```
package org.vitapasser.photocopypoint.Model;

import java.util.Objects;

public record Money(Double value, String unit) {
    public Money sum(Money money) {
        try {
            if (!Objects.equals(unit, money.unit())){
                throw new Exception("Money units don't match");
            }
            return new Money(value + money.value(), unit);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}

@Override
public String toString() {
    return String.format("%.2f", value) + " " + unit;
}
}
```

Operator.java

```
package org.vitapasser.photocopypoint.Model;

public record Operator(Long id, String name) {}
```

Order.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public class Order {
```

```
    private List<TypeItem> typeItems;
```

```
    private Term term;
```

```
    private Money price;
```

```
    private String fullName;
```

```
    private String phoneNumber;
```

```
    private Operator operator;
```

```
    private Order() {}
```

```
    public static Order create() {
```

```
        Order order = new Order();
```

```
        order.typeItems = new ArrayList<>();
```

```
        return order;
```

```
    }
```

```
    public void addType(String name, TypeList typeList, int count) {
```

```
        Type type = typeList.getType(name);
```

```
        typeItems.add(new TypeItem(type, count));
```

```
    }
```

```
    public Term getTerm() {
```

```
        term = typeItems.stream()
```

```
            .map(TypeItem::getTerm)
```

```

        .reduce(Term::sum)
        .orElseThrow();
    return term;
}

public Money getPrice() {
    return typeItems.stream()
        .map(TypeItem::getMoney)
        .reduce(Money::sum)
        .orElseThrow();
}

public Money makePayment(Money countPayMoney,
                        String fullName,
                        String phoneNumber,
                        PickupStation pickUpStation,
                        TicketList ticketList) {
    this.fullName = fullName;
    this.phoneNumber = phoneNumber;
    operator = pickUpStation.getOperator();

    OrderIDAndOddMany orderIDAndOddMoney = pickUpStation.fixSale(this,
countPayMoney);
    assert orderIDAndOddMoney != null;
    Money change = orderIDAndOddMoney.oddMoney();
    ticketList.createTicket(orderIDAndOddMoney.orderID());
    return change;
}

/*
* Якщо потрібно буде отримати данні замовлення для інших програмних
потреб.

```

```
* */  
  
public List<TypeItem> getTypeItems() {  
    return typeItems;  
}  
  
public String getFullName() {  
    return fullName;  
}  
  
public String getPhoneNumber() {  
    return phoneNumber;  
}  
  
public Operator getOperator() {  
    return operator;  
}  
  
public void setFullName(String fullName) {  
    this.fullName = fullName;  
}  
  
public void setPhoneNumber(String phoneNumber) {  
    this.phoneNumber = phoneNumber;  
}  
  
public void setOperator(Operator operator) {  
    this.operator = operator;  
}  
}
```

OrderIDAndOddMany.java

```
package org.vitapasser.photocopypoint.Model;
```

```
public record OrderIDAndOddMany(long orderID, Money oddMoney) {}
```

OrderList.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import org.vitapasser.photocopypoint.Controller.DTO.OrderView;
```

```
import org.vitapasser.photocopypoint.Controller.TakeMoreInfo.TypeView;
```

```
import org.vitapasser.photocopypoint.Util.Mysql;
```

```
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public record OrderList(Connection connectionToDataBase) {
```

```
    public OrderViewAndTypesViews getFullInfoAboutOrder(long ticketId) {
```

```
        try {
```

```
            String sql = ""
```

```
                SELECT T.id as TicketId,
```

```
                    O.id as OrderId,
```

```
                    SCi.full_name as StaffFullName,
```

```
                    SCi.phone_number as StaffPhoneNumber,
```

```
                    CCi.full_name as ClientFullName,
```

```
                    CCi.phone_number as ClientPhoneNumber,
```

```
                    OS.specialization as StaffSpecialization,
```

S.address as AddressMake,
 M.count as CostCount,
 M.unit as CostUnit,
 C.station_name as ClientBeginOrderStationName,
 TSO.count as TypeServiceCount,
 TSO.term as TypeServiceOrderTerm,
 TS.info as TypeServiceInfo,
 TS.name as TypeServiceName,
 TS.term as TypeServiceTerm,
 TS.create_time as TypeCreateTime,
 TS.id as TypeId,
 T.isReady as isMadeOrder,
 MP.count as TypePriceCount,
 MP.unit as TypePriceUnit

From PhotocopyPoint.Ticket T

```

    inner join PhotocopyPoint.`Order` O on O.id = T.order_id
    inner join PhotocopyPoint.OrderStaff OS on O.id = OS.order_id
    inner join PhotocopyPoint.Staff S on OS.staff_id = S.id
    inner join PhotocopyPoint.Money M on O.money_id = M.id
    inner join PhotocopyPoint.Client C on O.client_id = C.id
    inner join PhotocopyPoint.Contact_info SCi on SCi.id =
S.contact_info_id
    inner join PhotocopyPoint.Contact_info CCI on CCI.id =
C.contact_info_id
    inner join PhotocopyPoint.TypeServiceOrder TSO on O.id =
TSO.order_id
    inner join PhotocopyPoint.TypeService TS on TSO.type_service_id
= TS.id
    inner join PhotocopyPoint.Price P on TS.price_id = P.id
    inner join PhotocopyPoint.Money MP on P.money_id = MP.id
  
```

```

where T.id = ?
group by TicketId,\s
        OrderId,\s
        StaffFullName,\s
        StaffPhoneNumber,\s
        ClientFullName,\s
        ClientPhoneNumber,
        StaffSpecialization,
        AddressMake,
        CostCount,
        CostUnit,
        ClientBeginOrderStationName,
        TypeServiceCount,
        TypeServiceOrderTerm,
        TypeServiceInfo,
        TypeServiceName,
        TypeServiceTerm,
        TypeCreateTime,
        TypeId,
        TicketId,
        isMadeOrder,
        TypePriceCount,
        TypePriceUnit""";

```

```

PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);
statement.setLong(1, ticketId);
statement.execute();

ResultSet sqlResult = statement.getResultSet();

```

```

sqlResult.next();

var orderId1 = sqlResult.getLong("OrderId");
var ticketId1 = sqlResult.getLong("TicketId");
var isReady = sqlResult.getBoolean("isMadeOrder");
var clientFullName = sqlResult.getString("ClientFullName");
var clientPhoneNumber = sqlResult.getString("ClientPhoneNumber");
var staffFullName = sqlResult.getString("StaffFullName");
var staffPhoneNumber = sqlResult.getString("StaffPhoneNumber");
var staffSpecialization = sqlResult.getString("StaffSpecialization");
var addressMake = sqlResult.getString("AddressMake");
var costCount = sqlResult.getDouble("CostCount");
var costUnit = sqlResult.getString("CostUnit");
var stationName = sqlResult.getString("ClientBeginOrderStationName");
var orderTerm = sqlResult.getString("TypeServiceOrderTerm");

List<TypeItem> typeItems = new ArrayList<>();
typeItems.add(new TypeItem(new Type(
    sqlResult.getLong("TypeId"),
    sqlResult.getString("TypeServiceName"),
    sqlResult.getString("TypeServiceInfo"),
    new Term(sqlResult.getTime("TypeServiceTerm")
        .toLocalTime()
        .toSecondOfDay()),
    new Money(
        sqlResult.getDouble("TypePriceCount"),
        sqlResult.getString("TypePriceUnit")
    ),
    ),

```



```

Mysql.dbDateTimeToLocalDateTime(sqlResult.getString("TypeCreateTime")),
    sqlResult.getInt("TypeServiceCount"))
);
while (sqlResult.next()) {
    typeItems.add(new TypeItem(new Type(
        sqlResult.getLong("TypeId"),
        sqlResult.getString("TypeServiceName"),
        sqlResult.getString("TypeServiceInfo"),
        new Term(sqlResult.getTime("TypeServiceTerm")
            .toLocalTime()
            .toSecondOfDay()),
        new Money(
            sqlResult.getDouble("TypePriceCount"),
            sqlResult.getString("TypePriceUnit")
        ),

```

```

Mysql.dbDateTimeToLocalDateTime(sqlResult.getString("TypeCreateTime")),
    sqlResult.getInt("TypeServiceCount"))
);
}

```

```

OrderView orderView = new OrderView(
    orderId1,
    ticketId1,
    "Замовлення готово: " + String.valueOf(isReady),
    "ФІО клієнта: " + clientFullName,
    "Номер телефону клієнта: " + clientPhoneNumber,
    "ФІО оператора: " + staffFullName,
    "Номер телефону оператора: " + staffPhoneNumber,

```

```

        "Спеціалізація оператора: " + staffSpecialization,
        "Адреса створення: " + addressMake,
        "Ціна: " + String.valueOf(costCount) + costUnit,
        "Назва станції, де замовили: " + stationName,
        "Термін створення замовлення: " + orderTerm);

```

```

        return new OrderViewAndTypesViews(orderView,
typeItems.stream().map(TypeView::new).toList());

```

```

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    return null;
}
}

```

OrderViewAndTypesViews.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import org.vitapasser.photocopypoint.Controller.DTO.OrderView;
import org.vitapasser.photocopypoint.Controller.TakeMoreInfo.TypeView;
```

```
import java.util.List;
```

```

public record OrderViewAndTypesViews(OrderView orderView, List<TypeView>
typeViews) {
    }

```

PickUpStation.java

```

package org.vitapasser.photocopypoint.Model;

import java.sql.*;
import java.util.Objects;

public record PickUpStation(Connection connectionToDataBase, String address) {

    /*
     * Через запит до бази даних отримуються вільний оператор
     * та повертається методом. (return null - є заглушка)
     */
    public Operator getOperator() {
        try {
            Statement statement = connectionToDataBase.createStatement();

            ResultSet sqlResult = statement.executeQuery(
                "SELECT                                     PhotocopyPoint.Contact_info.id,
PhotocopyPoint.Contact_info.full_name\n" +
                "From PhotocopyPoint.Contact_info \n" +
                "inner      join      PhotocopyPoint.Staff      on
PhotocopyPoint.Staff.contact_info_id " +
                "= PhotocopyPoint.Contact_info.id \n" +
                "inner join PhotocopyPoint.PickUpStationStaff on " +
                "PhotocopyPoint.PickUpStationStaff.staff_id      =
PhotocopyPoint.Staff.id\n" +
                "left join PhotocopyPoint.OrderStaff on " +
                "PhotocopyPoint.OrderStaff.staff_id = ALL(" +
                "Select PhotocopyPoint.Staff.id from PhotocopyPoint.Staff " +
                "inner join PhotocopyPoint.PickUpStation " +

```

```

        "on PhotocopyPoint.PickUpStation.id = " +
        "PhotocopyPoint.PickUpStationStaff.pickUpStation_id " +
        "WHERE      PhotocopyPoint.PickUpStation.address      =
\"" + address + "\" )\n" +
        "left join PhotocopyPoint.`Order` on PhotocopyPoint.`Order`.id =
" +
        "PhotocopyPoint.OrderStaff.order_id\n" +
        "left join PhotocopyPoint.TypeServiceOrder " +
        "on      PhotocopyPoint.TypeServiceOrder.id      =
PhotocopyPoint.`Order`.`id`\n" +
        "where (PhotocopyPoint.TypeServiceOrder.term < NOW() or " +
        "(PhotocopyPoint.TypeServiceOrder.term is null)) LIMIT 1;");

        sqlResult.next();
        return new Operator(sqlResult.getLong("id"),
            sqlResult.getString("full_name"));

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    return null;
}

/*
 * Через запрос у базу даних отправляється замовлення.
 * Також метод рахує задачу. (return null - є заглушка)
 */
public OrderIDAndOddMany fixSale(Order order, Money countPayMoney) {
    try {

```

```

    long idMoney = insertMoney(order);
    long idContactInfo = insertContactInfo(order);
    long idClient = insertClient(idContactInfo);
    long idOrder = insertOrder(idMoney, idClient);
    order.getTypeItems().forEach(type -> {
        try {
            insertTypeServiceOrder(idOrder, type, order.getTerm());
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    });
    insertOrderStaff(idOrder, order.getOperator());

    Money oddMoney = null;
    Money orderPrice = order.getPrice();
    if (Objects.equals(orderPrice.unit(), countPayMoney.unit()))
        oddMoney = new Money(countPayMoney.value() - orderPrice.value(),
            orderPrice.unit());

    return new OrderIDAndOddMany(idOrder, oddMoney);

} catch (Exception e) {
    System.out.println(e.getMessage());
    e.printStackTrace();
}

return null;
}

```

```

private void insertOrderStaff(long idOrder, Operator operator) throws
SQLException {
    StringBuilder sql;
    PreparedStatement statement;
    sql = new StringBuilder("INSERT INTO
`PhotocopyPoint`.OrderStaff(order_id, staff_id, specialization) " +
        "VALUES (?, ?, 'Оператор принтеры');\n");
    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);
    statement.setLong(1, idOrder);
    statement.setLong(2, operator.id());

    checkAffectRows(statement.executeUpdate());
    getId(statement);
}

```

```

private void insertTypeServiceOrder(long idOrder, TypeItem type, Term term)
throws SQLException {
    StringBuilder sql;
    PreparedStatement statement;
    sql = new StringBuilder("INSERT INTO
`PhotocopyPoint`.TypeServiceOrder(order_id, term, type_service_id, " +
        "count) VALUES (?, ADDTIME(NOW(), SEC_TO_TIME(?)), ?, ?);");
    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);
    statement.setLong(1, idOrder);
    //statement.setLong(2, type.);
    statement.setInt(2, term.value());
    statement.setLong(3, type.type().id());
    statement.setInt(4, type.count());
}

```

```

        checkAffectRows(statement.executeUpdate());
        getId(statement);
    }

```

```

private long insertOrder(long idMoney, long idClient) throws SQLException {
    StringBuilder sql;
    PreparedStatement statement;
    sql = new StringBuilder("INSERT INTO `PhotocopyPoint`.`Order` (money_id,
client_id) " +
        "VALUES (?, ?);\n");
    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);
    statement.setLong(1, idMoney);
    statement.setLong(2, idClient);

    checkAffectRows(statement.executeUpdate());
    return getId(statement);
}

```

```

private long insertClient(long id) throws SQLException {
    StringBuilder sql;
    PreparedStatement statement;
    sql = new StringBuilder("INSERT INTO
`PhotocopyPoint`.Client(station_name, contact_info_id) " +
        "VALUES (?, ?);\n");
    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);
    statement.setString(1, address);
    statement.setLong(2, id);
}

```

```

        checkAffectRows(statement.executeUpdate());
        return getId(statement);
    }

```

```

private long insertContactInfo(Order order) throws SQLException {
    StringBuilder sql;
    PreparedStatement statement;

    sql = new StringBuilder("INSERT INTO
`PhotocopyPoint`.Contact_info(full_name, phone_number) " +
        "VALUES (?, ?);\n");

    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);

    statement.setString(1, order.getFullName());
    statement.setString(2, order.getPhoneNumber());

    checkAffectRows(statement.executeUpdate());
    return getId(statement);
}

```

```

private long insertMoney(Order order) throws SQLException {
    StringBuilder sql;
    PreparedStatement statement;

    sql = new StringBuilder("INSERT INTO `PhotocopyPoint`.Money(count,
unit) VALUES (?, ?);");

    statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);

    statement.setDouble(1, order.getPrice().value());
    statement.setString(2, order.getPrice().unit());
}

```



```

        checkAffectRows(statement.executeUpdate());
        return getId(statement);
    }

```

```

private static long getId(PreparedStatement statement) throws SQLException {
    try (ResultSet generatedKeys = statement.getGeneratedKeys()) {
        if (generatedKeys.next()) {
            return generatedKeys.getLong(1);
        }
        else {
            throw new SQLException("Creating failed, no ID obtained.");
        }
    }
}

```

```

private static void checkAffectRows(int affectedRows) throws SQLException {
    if (affectedRows == 0) {
        throw new SQLException("Creating failed, no rows affected.");
    }
}

```

Register.java

```
package org.vitapasser.photocopypoint.Model;
```

```

import org.vitapasser.photocopypoint.Controller.DTO.OrderView;
import org.vitapasser.photocopypoint.Exception.NotExistTypeException;

```

```
import java.util.List;
```

```
public class Register {  
    private Order order;  
    private final TypeList typeList;  
    private final PickupStation pickupStation;  
    private final TicketList ticketList;  
    private final OrderList orderList;  
  
    public Register(TypeList typeList, PickupStation pickupStation, TicketList  
ticketList, OrderList orderList) {  
        this.typeList = typeList;  
        this.pickupStation = pickupStation;  
        this.ticketList = ticketList;  
        this.orderList = orderList;  
    }  
  
    public void newOrder() {  
        this.order = Order.create();  
    }  
  
    public List<TypeItem> addType(String name, int count) {  
        order.addType(name, typeList, count);  
        return order.getTypeItems();  
    }  
  
    public Term getTerm() {  
        return order.getTerm();  
    }  
  
    public Money getPrice() {  
        return order.getPrice();  
    }  
}
```

```
}
```

```
public Money makePayment(Money countPayMoney,
                        String fullName,
                        String phoneNumber){
    return order.makePayment(countPayMoney,
                            fullName,
                            phoneNumber,
                            pickUpStation,
                            ticketList);
}
```

```
public List<Type> getTypes(String likeNameTypes) throws
NotExistTypeException {
    return typeList.getTypes(likeNameTypes);
}
```

```
public List<TypeItem> getOrderTypes() {
    return order.getTypeItems();
}
```

```
public List<Ticket> getAllTickets() {
    return ticketList.getAllTickets();
}
```

```
public List<Ticket> getAllMadeTickets() {
    return ticketList.getAllMadeTickets();
}
```

```
public OrderViewAndTypesViews getFullInfoAboutOrder(long TicketId) {
```

```

        return orderList.getFullInfoAboutOrder(TicketId);
    }

    public void giveTheOrder(long TicketId) {
        ticketList.setEndOrderTicket(TicketId);
    }

    public void madeTheOrder(long TicketId) {
        ticketList.setMadeOrderTicket(TicketId);
    }

    public List<Ticket> getAllNotMadeTickets(String fullNameStaff) {
        return ticketList.getAllNotMadeTickets(fullNameStaff);
    }
}

```

Term.java

```

package org.vitapasser.photocopypoint.Model;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public record Term(Integer value) {

    public Term sum(Term term) {
        return new Term(this.value() + term.value());
    }

    public LocalDateTime toLocalTime(){
        return LocalDateTime.ofSecondOfDay(value);
    }
}

```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    LocalDateTime time = toLocalTime();
```

```
    return time.format(DateTimeFormatter.ofPattern("HH:mm:ss"));
```

```
}
```

```
}
```

Ticket.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import java.util.Objects;
```

```
public final class Ticket {
```

```
    private long id;
```

```
    private long orderId;
```

```
    private String name;
```

```
    private boolean isReady;
```

```
    private boolean isDone;
```

```
    private String fullName;
```

```
    private String phoneNumber;
```

```
    public Ticket(long id, long orderId, String name, boolean isReady, boolean  
isDone, String fullName, String phoneNumber) {
```

```
        this.id = id;
```

```
        this.orderId = orderId;
```

```
        this.name = name;
```

```
        this.isReady = isReady;
```

```
    this.isDone = isDone;
    this.fullName = fullName;
    this.phoneNumber = phoneNumber;
}
```

```
public long getId() {
    return id;
}
```

```
public void setId(long id) {
    this.id = id;
}
```

```
public long getOrderId() {
    return orderId;
}
```

```
public void setOrderId(long orderId) {
    this.orderId = orderId;
}
```

```
public String getName() {
    return name;
}
```

```
public void setName(String name) {
    this.name = name;
}
```

```
public boolean getIsReady() {
```

```
        return isReady;
    }

    public void setIsReady(boolean ready) {
        isReady = ready;
    }

    public boolean getIsDone() {
        return isDone;
    }

    public void setIsDone(boolean done) {
        isDone = done;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}
```

```
}
```

TicketList.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import java.sql.*;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public record TicketList(Connection connectionToDataBase) {
```

```
    public void setEndOrderTicket(long id) {
```

```
        try {
```

```
            String sql = ""
```

```
                UPDATE PhotocopyPoint.Ticket PT
```

```
                SET PT.is_done = TRUE
```

```
                WHERE PT.id = ?;"";
```

```
            PreparedStatement
```

```
            statement
```

```
=
```

```
connectionToDataBase.prepareStatement(sql);
```

```
        statement.setLong(1, id);
```

```
        checkAffectRows(statement.executeUpdate());
```

```
    } catch (Exception e) {
```

```
        System.out.println(e.getMessage());
```

```
    }
```

```
}
```



```

public void setMadeOrderTicket(long id) {
    try {
        String sql = ""
            UPDATE PhotocopyPoint.Ticket PT
            SET PT.isReady = TRUE
            WHERE PT.id = ?;"";

        PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);
        statement.setLong(1, id);

        checkAffectRows(statement.executeUpdate());

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

}

public List<Ticket> getAllTickets() {
    try {
        String sql = ""
            SELECT PT.id,
                PT.order_id,
                GROUP_CONCAT(TS.name SEPARATOR ', ') AS all_names,
                PT.isReady,
                PT.is_done,
                Ci.full_name,
                Ci.phone_number
            FROM PhotocopyPoint.Ticket PT

```

```

        INNER JOIN PhotocopyPoint.`Order` PO ON PO.id =
PT.order_id

        INNER JOIN PhotocopyPoint.Client C ON PO.client_id = C.id
        INNER JOIN PhotocopyPoint.Contact_info Ci ON
C.contact_info_id = Ci.id

        INNER JOIN PhotocopyPoint.TypeServiceOrder TSO ON PO.id
= TSO.order_id

        INNER JOIN PhotocopyPoint.TypeService TS ON
TSO.type_service_id = TS.id

        GROUP BY PT.id, PT.order_id, PT.isReady, Ci.full_name,
Ci.phone_number

        ORDER BY PT.id ASC;
""";

```

```

        PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);

```

```

        statement.executeQuery();

```

```

        return getTickets(statement);

```

```

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

```

```

    return null;

```

```

}

```

```

public List<Ticket> getAllMadeTickets() {

```

```

try {
    String sql = ""
        SELECT PT.id,
            PT.order_id,
            GROUP_CONCAT(TS.name SEPARATOR ', ') AS all_names,
            PT.isReady,
            PT.is_done,
            Ci.full_name,
            Ci.phone_number
        FROM PhotocopyPoint.Ticket PT
            INNER JOIN PhotocopyPoint.`Order` PO ON PO.id =
PT.order_id
            INNER JOIN PhotocopyPoint.Client C ON PO.client_id = C.id
            INNER JOIN PhotocopyPoint.Contact_info Ci ON
C.contact_info_id = Ci.id
            INNER JOIN PhotocopyPoint.TypeServiceOrder TSO ON PO.id
= TSO.order_id
            INNER JOIN PhotocopyPoint.TypeService TS ON
TSO.type_service_id = TS.id
        where PT.isReady = TRUE and PT.is_done = FALSE
        GROUP BY PT.id, PT.order_id, PT.isReady, PT.is_done,
Ci.full_name, Ci.phone_number
        ORDER BY PT.id ASC;
    """;

    PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);

    statement.executeQuery();

```

```
return getTickets(statement);
```

```
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

```
return null;
```

```
}
```

```
public List<Ticket> getAllNotMadeTickets(String fullNameStaff) {
```

```
try {
```

```
    String sql = ""
```

```
        SELECT PT.id,
```

```
        PT.order_id,
```

```
        GROUP_CONCAT(TS.name SEPARATOR ', ') AS all_names,
```

```
        PT.isReady,
```

```
        PT.is_done,
```

```
        Ci.full_name,
```

```
        Ci.phone_number
```

```
    FROM PhotocopyPoint.Ticket PT
```

```
        INNER JOIN PhotocopyPoint.`Order` PO ON PO.id =
```

```
PT.order_id
```

```
        INNER JOIN PhotocopyPoint.Client C ON PO.client_id = C.id
```

```
        INNER JOIN PhotocopyPoint.Contact_info Ci ON
```

```
C.contact_info_id = Ci.id
```

```
        INNER JOIN PhotocopyPoint.TypeServiceOrder TSO ON PO.id
```

```
= TSO.order_id
```

```
        INNER JOIN PhotocopyPoint.TypeService TS ON
```

```
TSO.type_service_id = TS.id
```

```

        inner join PhotocopyPoint.OrderStaff OS on PO.id = OS.order_id
        inner join PhotocopyPoint.Staff S on OS.staff_id = S.id
        inner join (select SCI.full_name, SCI.id from
PhotocopyPoint.Contact_info SCI) as SCI on SCI.id = S.contact_info_id
        where PT.isReady = FALSE and PT.is_done = FALSE and
SCI.full_name = ?
        GROUP BY PT.id, PT.order_id, PT.isReady, PT.is_done,
Ci.full_name, Ci.phone_number
        ORDER BY PT.id ASC;
        """;

```

```

        PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);
        statement.setString(1, fullNameStaff);

        statement.executeQuery();

        return getTickets(statement);

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    return null;

}

```

```

private static List<Ticket> getTickets(PreparedStatement statement) throws
SQLException {
    ResultSet resultSet = statement.getResultSet();

```

```

List<Ticket> tickets = new ArrayList<>();

while(resultSet.next()){
    tickets.add(new Ticket(resultSet.getLong("id"),
        resultSet.getLong("order_id"),
        resultSet.getString("all_names"),
        resultSet.getBoolean("isReady"),
        resultSet.getBoolean("is_done"),
        resultSet.getString("full_name"),
        resultSet.getString("phone_number")));
}
return tickets;
}

public long createTicket(long idOrder) {
    long ticketId = 0;
    try {
        ticketId = insertTicket(idOrder);

        return ticketId;

    } catch (SQLException e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
    return ticketId;
}

private long insertTicket(long idOrder) throws SQLException {

```

```

        StringBuilder sql;
        PreparedStatement statement;
        sql = new StringBuilder("INSERT INTO `PhotocopyPoint`.Ticket(order_id)
VALUES (?);\n");
        statement = connectionToDataBase.prepareStatement(sql.toString(),
Statement.RETURN_GENERATED_KEYS);
        statement.setLong(1, idOrder);

        checkAffectRows(statement.executeUpdate());
        return getId(statement);
    }

    private static long getId(PreparedStatement statement) throws SQLException {
        try (ResultSet generatedKeys = statement.getGeneratedKeys()) {
            if (generatedKeys.next()) {
                return generatedKeys.getLong(1);
            }
            else {
                throw new SQLException("Creating failed, no ID obtained.");
            }
        }
    }

    private static void checkAffectRows(int affectedRows) throws SQLException {
        if (affectedRows == 0) {
            throw new SQLException("Creating failed, no rows affected.");
        }
    }
}

```

Type.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import java.time.LocalDateTime;
```

```
public record Type(Long id, String name, String info, Term term, Money money,
LocalDateTime create_time) {

}
```

TypeItem.java

```
package org.vitapasser.photocopypoint.Model;
```

```
public record TypeItem(Type type, int count) {

    public Money getMoney() {
        return new Money(type.money().value() * count, type.money().unit());
    }

    public Term getTerm() {
        return new Term(type.term().value() * count);
    }
}
```

TypeList.java

```
package org.vitapasser.photocopypoint.Model;
```

```
import org.vitapasser.photocopypoint.Exception.NotExistTypeException;
import org.vitapasser.photocopypoint.Util.Mysql;
```



```

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class TypeList {

    private final Connection connectionToDataBase;

    public TypeList(Connection connectionToDataBase) {
        this.connectionToDataBase = connectionToDataBase;
    }

    public List<Type> getTypes(String likeNameTypes) throws
NotExistTypeException {

        List<Type> types = new ArrayList<>();

        try {
            String sql = ""
                SELECT TS.name,
                    TS.id,
                    TS.info,
                    TS.term,
                    PM.count,
                    PM.unit,
                    TS.create_time
                FROM PhotocopyPoint.TypeService TS
                INNER JOIN (
                    SELECT name, MAX(create_time) AS max_create_time
                    FROM PhotocopyPoint.TypeService

```

```

        GROUP BY name
    ) AS latest_services
    ON TS.name = latest_services.name AND TS.create_time =
latest_services.max_create_time

    INNER JOIN PhotocopyPoint.Price PP ON TS.price_id = PP.id
    INNER JOIN PhotocopyPoint.Money PM ON PP.money_id = PM.id
    WHERE TS.name LIKE CONCAT('%', ?, '%')
    ORDER BY TS.id DESC;
    """;

```

```

        PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);

        statement.setString(1, likeNameTypes);

        statement.executeQuery();

        ResultSet resultSet = statement.getResultSet();

        while(resultSet.next()){
            types.add(new Type(resultSet.getLong("id"),
                resultSet.getString("name"),
                resultSet.getString("info"),
                new
Term(resultSet.getTime("term").toLocalTime().toSecondOfDay()),
                new Money(resultSet.getDouble("count"),
resultSet.getString("unit")),

            Mysql.dbDateTimeToLocalDateTime(resultSet.getString("create_time"))));
        }

```

```

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

```

```

    if (types.isEmpty()) {
        throw new NotExistTypeException("Помилкова назва послуги!");
    }

```

```

    return types;

```

```

}

```

```

public Type getType(String typeName) {

```

```

    try {

```

```

        String sql = ""

```

```

            SELECT TS.id,

```

```

                TS.name,

```

```

                TS.info,

```

```

                TS.term,

```

```

                PM.count,

```

```

                PM.unit,

```

```

                TS.create_time

```

```

            From PhotocopyPoint.Money PM

```

```

            inner join PhotocopyPoint.Price PP on PM.id = PP.money_id

```

```

            inner join PhotocopyPoint.TypeService TS on PP.id = TS.price_id

```

```

            where TS.name like CONCAT('%', ?, '%')

```

```

            ORDER BY TS.create_time DESC LIMIT 1;"";

```

```

        PreparedStatement

```

```

        statement

```

```

        =

```

```

        connectionToDataBase.prepareStatement(sql);

```

```

statement.setString(1, typeName);

statement.executeQuery();

ResultSet resultSet = statement.getResultSet();

resultSet.next();
return new Type(resultSet.getLong("id"),
    resultSet.getString("name"),
    resultSet.getString("info"),
    new
Term(resultSet.getTime("term").toLocalTime().toSecondOfDay()),
    new Money(resultSet.getDouble("count"), resultSet.getString("unit")),

Mysql.dbDateTimeToLocalDateTime(resultSet.getString("create_time")));

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    return null;
}
}

```

NotExistTypeException.java

```
package org.vitapasser.photocopypoint.Exception;
```

```

public class NotExistTypeException extends Exception{
    public NotExistTypeException(String message) {
        super(message);
    }
}

```

```

    }
}

```

Mysql.java

```
package org.vitapasser.photocopypoint.Util;
```

```

import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.time.LocalDateTime;
import java.util.Properties;

```

```

public class Mysql {
    public static LocalDateTime dbDateTimeToLocalDateTime(String dateTime){
        String dateTimeForParse = dateTime.replace(' ', 'T');
        return LocalDateTime.parse(dateTimeForParse);
    }

```

```
    public static Connection getConnection() throws SQLException, IOException {
```

```
        Properties props = new Properties();
```

```
        try(InputStream in =
```

```
Files.newInputStream(Paths.get("Configuration/database.properties"))){
```

```
            props.load(in);
```

```
        }
```

```
        String url = props.getProperty("url");
```

```
String username = props.getProperty("username");  
String password = props.getProperty("password");  
  
return DriverManager.getConnection(url, username, password);  
}  
}
```

ДОДАТОК Б КОД КОНТРОЛЕРІВ, ТОЧКУ ВХОДУ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ТА MODUL-INFO

PaymentController.java

```
package org.vitapasser.photocopypoint.Controller;
```

```
import javafx.application.Platform;
```

```
import javafx.collections.FXCollections;
```

```
import javafx.collections.ObservableList;
```

```
import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Node;
```

```
import javafx.scene.Scene;
```

```
import javafx.scene.control.*;
```

```
import javafx.scene.control.cell.PropertyValueFactory;
```

```
import javafx.scene.layout.VBox;
```

```
import javafx.stage.Stage;
```

```
import org.vitapasser.photocopypoint.Controller.OrderManagement.Controller;
```

```
import org.vitapasser.photocopypoint.MainApplication;
```

```
import org.vitapasser.photocopypoint.Model.Money;
```

```
import org.vitapasser.photocopypoint.Model.Register;
```

```
import org.vitapasser.photocopypoint.Model.Ticket;
```

```
import java.io.IOException;
```

```
import java.util.List;
```

```
import java.util.Objects;
```

```
public class PaymentController {
```

```
    private Register register;
```

String nameClient;

String phoneNumberClient;

ObservableList<Ticket> ticketsList = FXCollections.observableArrayList();

@FXML

TableView<Ticket> listOfTicketsTableView;

@FXML

private TableColumn<Ticket, String> fullNameColumn;

@FXML

private TableColumn<Ticket, Long> idTicketColumn;

@FXML

private TableColumn<Ticket, Boolean> isReadyColumn;

@FXML

private TableColumn<Ticket, String> namesTypeServiceColumn;

@FXML

private TableColumn<Ticket, String> phoneNumberColumn;

@FXML

Label resultOddMoney;

@FXML

Label resultPrice;


```
@FXML
```

```
Label payLabel;
```

```
@FXML
```

```
TextField countClientPayTextField;
```

```
@FXML
```

```
VBox oddPayVBox;
```

```
@FXML
```

```
Button acceptButton;
```

```
@FXML
```

```
Button cancelButton;
```

```
@FXML
```

```
private void onPaymentButtonClick(ActionEvent event) {
```

```
    if (countClientPayTextField.getText().isEmpty()) {
```

```
        payLabel.setText("До сплати| Введіть суму оплаченою клієнтом");
```

```
        return;
```

```
    }
```

```
    try {
```

```
        if (Double.parseDouble(countClientPayTextField.getText())
```

```
<
```

```
register.getPrice().value()){
```

```
        payLabel.setText("До сплати| Не вистачає\n грошей для оплати!");
```

```
        return;
```

```
    } else {
```

```

        payLabel.setText("До сплати");
    }

    oddPayVBox.setDisable(false);

    resultOddMoney.setText(register.makePayment(
        new Money(Double.parseDouble(countClientPayTextField.getText()),
register.getPrice().unit()),
        nameClient,
        phoneNumberClient).toString());

    cancelButton.setDisable(true);
    acceptButton.setDisable(true);
    changeTable();
} catch (NumberFormatException e) {
    payLabel.setText("До сплати| Використовуйте\n тільки числа та
точку!");
}
}

@FXML
private void onOrderManagementButtonClick(ActionEvent event) throws
IOException {

    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("order-management.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    Controller controller = FXMLLoader.getController();
    controller.putData(register);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

```

```

stage.setScene(scene);
stage.setTitle("Створення нового замовлення");
stage.show();
}

```

@FXML

```
protected void onCancelButtonClick(ActionEvent event) throws IOException {
```

```

    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("creating-order.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.CreatingOrder.Controller controller
= FXMLLoader.getController();
    controller.putData(register);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Менеждер замовлень");
    stage.show();
}

```

```

private void changeTable() {
    List<Ticket> tickets = register.getAllTickets();

    ticketsList.clear();
    ticketsList.addAll(tickets);
}

```

@FXML

```

private void initialize() {
    Platform.runLater(() -> {

```

```
resultPrice.setText(register.getPrice().toString());
```

```
listOfTicketsTableView.setItems(ticketsList);
```

```
ticketsList.clear();
```

```
idTicketColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
```

```
fullNameColumn.setCellValueFactory(new  
PropertyValueFactory<>("fullName"));
```

```
phoneNumberColumn.setCellValueFactory(new  
PropertyValueFactory<>("phoneNumber"));
```

```
isReadyColumn.setCellValueFactory(new  
PropertyValueFactory<>("isReady"));
```

```
namesTypeServiceColumn.setCellValueFactory(new  
PropertyValueFactory<>("name"));
```

```
changeTable();
```

```
});
```

```
}
```

```
public void putData(Register register, String nameClient, String  
phoneNumberClient) {
```

```
    this.register = register;
```

```
    this.nameClient = nameClient;
```

```
    this.phoneNumberClient = phoneNumberClient;
```

```
}
```

```
}
```

CreatingOrder/Controller.java

```
package org.vitapasser.photocopypoint.Controller.CreatingOrder;

import com.mysql.cj.util.StringUtils;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import org.vitapasser.photocopypoint.Controller.PaymentController;
import org.vitapasser.photocopypoint.Controller.DTO.TypeView;
import org.vitapasser.photocopypoint.Exception.NotExistTypeException;
import org.vitapasser.photocopypoint.MainApplication;
import org.vitapasser.photocopypoint.Model.Register;
import org.vitapasser.photocopypoint.Model.Type;
import org.vitapasser.photocopypoint.Model.TypeItem;

import java.io.IOException;
import java.time.LocalDateTime;
import java.util.List;
```

```
import java.util.Objects;
```

```
public class Controller {
```

```
        ObservableList<TypeView>        typeViewsListOfSelectableServices        =
FXCollections.observableArrayList();
```

```
        ObservableList<TypeView>        typeViewsListSelectedServices        =
FXCollections.observableArrayList();
```

```
        Register register;
```

```
        @FXML
```

```
        private TableColumn<TypeView, Double> countMoneyTypeColumn;
```

```
        @FXML
```

```
        private TableColumn<TypeView, Double> countMoneyTypeColumn1;
```

```
        @FXML
```

```
        private TableColumn<TypeView, String> currencyMoneyTypeColumn;
```

```
        @FXML
```

```
        private TableColumn<TypeView, String> currencyMoneyTypeColumn1;
```

```
        @FXML
```

```
        private TableColumn<TypeView, Long> idTypeColumn;
```

```
        @FXML
```

```
        private TableColumn<TypeView, Long> idTypeColumn1;
```

```
        @FXML
```

```
        private TableColumn<TypeView, String> infoTypeColumn;
```

@FXML

private TableColumn<TypeView, String> infoTypeColumn1;

@FXML

private TableColumn<TypeView, String> nameTypeColumn;

@FXML

private TableColumn<TypeView, String> nameTypeColumn1;

@FXML

private TableColumn<TypeView, LocalTime> termTypeColumn;

@FXML

private TableColumn<TypeView, String> termTypeColumn1;

@FXML

protected TableColumn<TypeView, Integer> countTypeColumn1;

@FXML

protected TextField nameTypeService;

@FXML

protected TextField countTypeService;

@FXML

protected VBox clientVBox;

@FXML

protected TextField nameClient;

@FXML

protected TextField phoneNumberClient;

@FXML

protected Label resultPrice;

@FXML

protected Label resultTerm;

@FXML

protected Label nameNameTypeService;

@FXML

protected TableView<TypeView> listSelectedServices;

@FXML

protected TableView<TypeView> listOfSelectableServices;

@FXML

protected Button paymentButton;

@FXML

protected Button addTypeServiceButton;

@FXML

protected Label countTypeServiceLabel;

@FXML

protected Label phoneNumberLabel;

@FXML

```
protected void onChangeNameTypeServiceTextField(KeyEvent event) {
    changeTable(typeViewsListOfSelectableServices);
}
```

@FXML

```
protected void onGetNameTypeServiceTableView(MouseEvent event) {
    TypeView typeView =
listOfSelectableServices.getSelectionModel().getSelectedItem();
    if (typeView == null) return;
    nameTypeService.setText(typeView.getName());
}
```

@FXML

```
protected void onWriteCheckTextField(KeyEvent event) {
    countTypeServiceLabel.setText("Кількість послуги");
    if (!StringUtils.isStrictlyNumeric(countTypeService.getText())) {
        addTypeServiceButton.setDisable(true);
        countTypeServiceLabel.setText("Кількість послуги| Введіть тільки
числове значення!");
    } else {
        addTypeServiceButton.setDisable(false);
    }
}
```

```
private void changeTable(ObservableList<TypeView> list) {
    List<Type> types;
    nameNameTypeService.setText("Назва послуги");
}
```

```

addTypeServiceButton.setDisable(false);
try {
    types = register.getTypes(nameTypeService.getText());

    list.clear();
    types.forEach(type -> list.add(new TypeView(type)));
} catch (NotExistTypeException e) {
    String string = "Назва послуги| " + e.getMessage();
    nameNameTypeService.setText(string);
    addTypeServiceButton.setDisable(true);
}
}

private void initLoadTackedTable() {
    List<TypeItem> types = register.getOrderTypes();

    typeViewsListSelectedServices.clear();
    types.forEach(type -> typeViewsListSelectedServices.add(new
TypeView(type)));
}

@FXML
protected void onAddTypeServiceButtonClick(ActionEvent event) {
    if (nameTypeService.getText().isEmpty() ||
countTypeService.getText().isEmpty()) {
        return;
    }

    List<TypeItem> typeItem = register.addType(nameTypeService.getText(),
Integer.parseInt(countTypeService.getText()));

```

```

typeViewsListSelectedServices.clear();
typeItem.forEach(typeItem1 -> typeViewsListSelectedServices.add(new
TypeView(typeItem1)));
clientVBox.setDisable(false);

resultPrice.setText(register.getPrice().toString());
resultTerm.setText(register.getTerm().toString());
}

```

@FXML

```

protected void onWriteNamePhoneNumber(KeyEvent event) {
    if (!phoneNumberClient.getText().matches("^(\\+\\d{1,3}\\s)?1?\\-
?\\.?\\s?(\\d{3})?([\\s.-]?\\d{3}[\\s.-]?\\d{4}$")){
        phoneNumberLabel.setText("Номер телефону клієнта| Помилковий
вираз!");
        paymentButton.setDisable(true);
        return;
    }
    phoneNumberLabel.setText("Номер телефону клієнта");

    paymentButton.setDisable(false);

}

```

```

private boolean isEmptyNamePhoneNumberClient() {
    return nameClient.getText().isEmpty() ||
phoneNumberClient.getText().isEmpty();
}

```

@FXML

```
protected void onPaymentButtonClick(ActionEvent event) throws IOException {
    if (isEmptyNamePhoneNumberClient()) {
        return;
    }
}
```

```
FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
    MainApplication.class.getResource("payment.fxml")));
```

```
Scene scene = new Scene(FXMLLoader.load());
```

```
PaymentController controller = FXMLLoader.getController();
```

```
controller.putData(register, nameClient.getText(),
```

```
phoneNumberClient.getText());
```

```
Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
```

```
stage.setScene(scene);
```

```
stage.setTitle("Оплата заовлення");
```

```
stage.show();
```

```
}
```

@FXML

```
protected void onCancelButtonClick(ActionEvent event) throws IOException {
```

```
FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
    MainApplication.class.getResource("order-management.fxml")));
```

```
Scene scene = new Scene(FXMLLoader.load());
```

```
org.vitapasser.photocopypoint.Controller.OrderManagement.Controller
```

```
controller = FXMLLoader.getController();
```

```
controller.putData(register);
```

```
Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
```

```
stage.setScene(scene);
```

```
stage.setTitle("Менеждер заовлень");
```

```
stage.show();
```

```
}
```

```

public void initialize() {
    Platform.runLater(() -> {
        listOfSelectableServices.setItems(typeViewsListOfSelectableServices);
        listSelectedServices.setItems(typeViewsListSelectedServices);
        typeViewsListOfSelectableServices.clear();
        typeViewsListSelectedServices.clear();

        idTypeColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
        idTypeColumn1.setCellValueFactory(new PropertyValueFactory<>("id"));

        nameTypeColumn.setCellValueFactory(new
Property ValueFactory<>("name"));
        nameTypeColumn1.setCellValueFactory(new
Property ValueFactory<>("name"));

        infoTypeColumn.setCellValueFactory(new
Property ValueFactory<>("info"));
        infoTypeColumn1.setCellValueFactory(new
Property ValueFactory<>("info"));

        termTypeColumn.setCellValueFactory(new
Property ValueFactory<>("term"));
        termTypeColumn1.setCellValueFactory(new
Property ValueFactory<>("term"));

        countMoneyTypeColumn.setCellValueFactory(new
Property ValueFactory<>("countMoney"));
        countMoneyTypeColumn1.setCellValueFactory(new
Property ValueFactory<>("countMoney"));
    });
}

```

```

        currencyMoneyTypeColumn.setCellValueFactory(new
PropertyVaultFactory<>("currencyMoney"));
        currencyMoneyTypeColumn1.setCellValueFactory(new
PropertyVaultFactory<>("currencyMoney"));

        countTypeColumn1.setCellValueFactory(new
PropertyVaultFactory<>("count"));

        changeTable(typeViewsListOfSelectableServices);
        initLoadTackedTable();

    });

}

public void putData(Register register) {
    this.register = register;
}
}

```

DTO/OrderView.java

```

package org.vitapasser.photocopypoint.Controller.DTO;

public record OrderView (
    long idOrder,
    long idTicket,
    String isReady,
    String clientFullName,
    String clientPhoneNumber,
    String staffFullName,
    String staffPhoneNumber,

```

```

    String staffSpecialization,
    String addressMake,
    String money,
    String stationName,
    String term
  ){}

```

DTO/TypeView.java

```
package org.vitapasser.photocopypoint.Controller.DTO;
```

```
import org.vitapasser.photocopypoint.Model.Type;
```

```
import org.vitapasser.photocopypoint.Model.TypeItem;
```

```
import java.time.LocalDateTime;
```

```
public class TypeView {
```

```
    long id;
```

```
    String name;
```

```
    String info;
```

```
    LocalDateTime term;
```

```
    Double countMoney;
```

```
    String currencyMoney;
```

```
    int count;
```

```
    public int getCount() {
```

```
        return count;
```

```
    }
```

```
    public void setCount(int count) {
```

```
        this.count = count;
```

```
}
```

```
public TypeView (Type type){
    this.id = type.id();
    this.name = type.name();
    this.info = type.info();
    this.term = type.term().toLocalTime();
    this.countMoney = type.money().value();
    this.currencyMoney = type.money().unit();
    this.count = 1;
}
```

```
public TypeView (TypeItem type){
    this.id = type.type().id();
    this.name = type.type().name();
    this.info = type.type().info();
    this.term = type.getTerm().toLocalTime();
    this.countMoney = type.getMoney().value();
    this.currencyMoney = type.getMoney().unit();
    this.count = type.count();
}
```

```
public TypeView(TypeItem type, boolean isForTakeMoreInfo){
    this.id = type.type().id();
    this.name = type.type().name();
    this.info = type.type().info();
    this.term = type.type().term().toLocalTime();
    this.countMoney = type.getMoney().value();
    this.currencyMoney = type.getMoney().unit();
    this.count = type.count();
}
```



```
}
```

```
public long getId() {  
    return id;  
}
```

```
public void setId(long id) {  
    this.id = id;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getInfo() {  
    return info;  
}
```

```
public void setInfo(String info) {  
    this.info = info;  
}
```

```
public LocalTime getTerm() {  
    return term;  
}
```

```

public void setTerm(LocalTime term) {
    this.term = term;
}

public Double getCountMoney() {
    return countMoney;
}

public void setCountMoney(Double countMoney) {
    this.countMoney = countMoney;
}

public String getCurrencyMoney() {
    return currencyMoney;
}

public void setCurrencyMoney(String currencyMoney) {
    this.currencyMoney = currencyMoney;
}
}

```

OrderManagement\Controller.java

```
package org.vitapasser.photocopypoint.Controller.OrderManagement;
```

```

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;

```

```

import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import org.vitapasser.photocopypoint.MainApplication;
import org.vitapasser.photocopypoint.Model.Register;
import org.vitapasser.photocopypoint.Model.Ticket;

import java.io.IOException;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Controller {

    ObservableList<Ticket> ticketsList = FXCollections.observableArrayList();

    private Register register;

    @FXML
    TableView<Ticket> listOfTicketsTableView;

    public final Runnable changeValuesOnListOfTicketsTableView = () -> {
        try {
            while (!Thread.currentThread().isInterrupted()) {

```

```

        int                index                =
listOfTicketsTableView.getSelectionModel().getFocusedIndex();

        List<Ticket> tickets = register.getAllMadeTickets();
        ticketsList.clear();
        ticketsList.addAll(tickets);
        listOfTicketsTableView.getSelectionModel().select(index);
        Thread.sleep(1000);
    }
} catch (Exception e) {
    throw new RuntimeException(e);
}
};

```

```

        private          final          ExecutorService          executor          =
Executors.newSingleThreadExecutor(runnable -> {
    Thread t = new Thread(runnable);
    t.setDaemon(true);
    return t ;
});

```

@FXML

```
private TableColumn<Ticket, String> fullNameColumn;
```

@FXML

```
private TableColumn<Ticket, Long> idTicketColumn;
```

@FXML

```
private TableColumn<Ticket, String> namesTypeServiceColumn;
```

@FXML

```
private TableColumn<Ticket, String> phoneNumberColumn;
private String fullNameStaff = "";
```

```
@FXML
```

```
protected void onStartCreateOrderButtonClick(ActionEvent event) throws
IOException {
    register.newOrder();

    this.executor.shutdown();
    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("creating-order.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.CreatingOrder.Controller controller
= FXMLLoader.getController();
    controller.putData(register);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Створення нового замовлення");
    stage.show();
}
```

```
@FXML
```

```
protected void onGetMoreNotMadeOrdersButtonClick(ActionEvent event)
throws IOException {

    this.executor.shutdown();
    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("take-orders.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
```

```

        org.vitapasser.photocopypoint.Controller.TakeOrders.Controller controller =
FXMLLoader.getController();
        controller.putData(register, fullNameStaff);
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.setTitle("Виконання існуючого замовлення");
        stage.show();
    }

```

@FXML

```

protected void onGetMoreInfoOrderButtonClick(ActionEvent event) throws
IOException {
    if (ticketsList.isEmpty()) return;

    this.executor.shutdown();

    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("take-more-info-order.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.TakeMoreInfo.Controller controller
= FXMLLoader.getController();
    controller.putData(register,
listOfTicketsTableView.getSelectionModel().getSelectedItem().getId());
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Оглядач готового замовлення");
    stage.show();
}

```

```

private void changeTable() {

```

```

    List<Ticket> tickets = register.getAllMadeTickets();

```

```

        ticketsList.clear();
        ticketsList.addAll(tickets);
    }
    public void initialize() {
        Platform.runLater(() -> {

            listOfTicketsTableView.setItems(ticketsList);
            ticketsList.clear();

            idTicketColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

            fullNameColumn.setCellValueFactory(new
PropertyCellValueFactory<>("fullName"));

            phoneNumberColumn.setCellValueFactory(new
PropertyCellValueFactory<>("phoneNumber"));

            namesTypeServiceColumn.setCellValueFactory(new
PropertyCellValueFactory<>("name"));

            changeTable();
            this.executor.submit(changeValuesOnListOfTicketsTableView);
        });
    }
    public void putData(Register register) {
        this.register = register;
    }
    public void putData(Register register, String fullNameStaff) {
        this.register = register;
    }

```

```

        this.fullNameStaff = fullNameStaff;
    }
}

```

TakeMoreInfo\Controller.java

```

package org.vitapasser.photocopypoint.Controller.TakeMoreInfo;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import org.vitapasser.photocopypoint.MainApplication;
import org.vitapasser.photocopypoint.Model.OrderViewAndTypesViews;
import org.vitapasser.photocopypoint.Model.Register;

import java.io.IOException;
import java.util.Objects;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Controller {
    private Register register;
    OrderViewAndTypesViews orderView;
}

```



```

    private long OrderID;

    private          ObservableList<String>          orderItems          =
FXCollections.observableArrayList();

    private          ObservableList<TypeView>        typeServices        =
FXCollections.observableArrayList();

```

```
@FXML
```

```
private TableColumn<TypeView, String> costTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> countTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> idTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> infoTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> nameTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> termTypeService;
```

```
@FXML
```

```
private ListView<String> listInfo;
```

```
@FXML
```

```
private TableView<TypeView> typeTableView;
```

@FXML

```
protected void onSetGiveTheOrderButtonClick(ActionEvent event) throws
IOException {
    register.giveTheOrder(orderView.orderView().idTicket());
    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("order-management.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.OrderManagement.Controller
controller = FXMLLoader.getController();
    controller.putData(register);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Менеждер заовленъ");
    stage.show();
}
```

@FXML

```
protected void onOrderManagementButtonClick(ActionEvent event) throws
IOException {
    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("order-management.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.OrderManagement.Controller
controller = FXMLLoader.getController();
    controller.putData(register);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Менеждер заовленъ");
    stage.show();
}
```

```

public void initialize() {
    Platform.runLater(() -> {
        orderView = register.getFullInfoAboutOrder(OrderID);
        orderItems.add("Ідентифікатор          замовлення:          "          +
String.valueOf(orderView.orderView().idOrder()));
        orderItems.add("Ідентифікатор          квитка:          "          +
String.valueOf(orderView.orderView().idTicket()));
        orderItems.add(orderView.orderView().isReady());
        orderItems.add(orderView.orderView().clientFullName());
        orderItems.add(orderView.orderView().clientPhoneNumber());
        orderItems.add(orderView.orderView().staffFullName());
        orderItems.add(orderView.orderView().staffPhoneNumber());
        orderItems.add(orderView.orderView().staffSpecialization());
        orderItems.add(orderView.orderView().addressMake());
        orderItems.add(orderView.orderView().money());
        orderItems.add(orderView.orderView().term());
        orderItems.add(orderView.orderView().stationName());
        listInfo.setItems(orderItems);
        typeServices.clear();
        typeServices.addAll(orderView.typeViews());

        idTypeService.setCellValueFactory(new PropertyValueFactory<>("id"));

        nameTypeService.setCellValueFactory(new
PropertyValueFactory<>("name"));

        infoTypeService.setCellValueFactory(new
PropertyValueFactory<>("info"));

```

```

        termTypeService.setCellValueFactory(new
PropertyCellValueFactory<>("term"));

        countTypeService.setCellValueFactory(new
PropertyCellValueFactory<>("count"));

        costTypeService.setCellValueFactory(new
PropertyCellValueFactory<>("money"));
        typeTableView.setItems(typeServices);
    });
}

public void putData(Register register, long OrderID) {
    this.register = register;
    this.OrderID = OrderID;
}
}

```

TakeMoreInfo\TypeView.java

```
package org.vitapasser.photocopypoint.Controller.TakeMoreInfo;
```

```
import org.vitapasser.photocopypoint.Model.Type;
import org.vitapasser.photocopypoint.Model.TypeItem;
```

```
import java.time.LocalTime;
```

```
public class TypeView {
    String id;
    String name;
    String info;
    String term;

```

String money;

String count;

```
public TypeView(TypeItem type){  
    this.id = type.type().id().toString();  
    this.name = type.type().name();  
    this.info = type.type().info();  
    this.term = type.type().term().toLocalTime().toString();  
    this.money = type.getMoney().toString();  
    this.count = String.valueOf(type.count());  
}
```

```
public String getCount() {  
    return count;  
}
```

```
public void setCount(String count) {  
    this.count = count;  
}
```

```
public String getMoney() {  
    return money;  
}
```

```
public void setMoney(String money) {  
    this.money = money;  
}
```

```
public String getTerm() {  
    return term;  
}
```

```
}

public void setTerm(String term) {
    this.term = term;
}

public String getInfo() {
    return info;
}

public void setInfo(String info) {
    this.info = info;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}
}
```

TakeMoreInfoForMake\Controller.java

```
package org.vitapasser.photocopypoint.Controller.TakeMoreInfoForMake;
```

```
import javafx.application.Platform;
```

```
import javafx.collections.FXCollections;
```

```
import javafx.collections.ObservableList;
```

```
import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Node;
```

```
import javafx.scene.Scene;
```

```
import javafx.scene.control.ListView;
```

```
import javafx.scene.control.TableColumn;
```

```
import javafx.scene.control TableView;
```

```
import javafx.scene.control.cell.PropertyValueFactory;
```

```
import javafx.stage.Stage;
```

```
import org.vitapasser.photocopypoint.Controller.TakeMoreInfo.TypeView;
```

```
import org.vitapasser.photocopypoint.MainApplication;
```

```
import org.vitapasser.photocopypoint.Model.OrderViewAndTypesViews;
```

```
import org.vitapasser.photocopypoint.Model.Register;
```

```
import java.io.IOException;
```

```
import java.util.Objects;
```

```
public class Controller {
```

```
    private Register register;
```

```
    OrderViewAndTypesViews orderView;
```

```
    private long OrderID;
```

```
    private String fullNameStaff;
```

```

        private          ObservableList<String>          orderItems          =
FXCollections.observableArrayList();
        private          ObservableList<TypeView>        typeServices        =
FXCollections.observableArrayList();

```

```
@FXML
```

```
private TableColumn<TypeView, String> costTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> countTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> idTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> infoTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> nameTypeService;
```

```
@FXML
```

```
private TableColumn<TypeView, String> termTypeService;
```

```
@FXML
```

```
private ListView<String> listInfo;
```

```
@FXML
```

```
private TableView<TypeView> typeTableView;
```

```
@FXML
```



```

        protected void onSetGiveTheOrderButtonClick(ActionEvent event) throws
IOException {
            register.madeTheOrder(orderView.orderView().idTicket());
            FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
                MainApplication.class.getResource("order-management.fxml")));
            Scene scene = new Scene(FXMLLoader.load());
            org.vitapasser.photocopypoint.Controller.OrderManagement.Controller
controller = FXMLLoader.getController();
            controller.putData(register);
            Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            stage.setScene(scene);
            stage.setTitle("Менеждер замовлень");
            stage.show();
        }

```

@FXML

```

        protected void onTakeOrderButtonClick(ActionEvent event) throws IOException
{
            FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
                MainApplication.class.getResource("take-orders.fxml")));
            Scene scene = new Scene(FXMLLoader.load());
            org.vitapasser.photocopypoint.Controller.TakeOrders.Controller controller =
FXMLLoader.getController();
            controller.putData(register, fullNameStaff);
            Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            stage.setScene(scene);
            stage.setTitle("Виконання існуючого замовлення");
            stage.show();
        }

        public void initialize() {

```

```

Platform.runLater(() -> {
    orderView = register.getFullInfoAboutOrder(OrderID);
    orderItems.add("Ідентифікатор          замовлення:          "          +
String.valueOf(orderView.orderView().idOrder()));
    orderItems.add("Ідентифікатор          квитка:          "          +
String.valueOf(orderView.orderView().idTicket()));
    orderItems.add(orderView.orderView().isReady());
    orderItems.add(orderView.orderView().clientFullName());
    orderItems.add(orderView.orderView().clientPhoneNumber());
    orderItems.add(orderView.orderView().staffFullName());
    orderItems.add(orderView.orderView().staffPhoneNumber());
    orderItems.add(orderView.orderView().staffSpecialization());
    orderItems.add(orderView.orderView().addressMake());
    orderItems.add(orderView.orderView().money());
    orderItems.add(orderView.orderView().term());
    orderItems.add(orderView.orderView().stationName());
    listInfo.setItems(orderItems);
    typeServices.clear();
    typeServices.addAll(orderView.typeViews());

    idTypeService.setCellValueFactory(new PropertyValueFactory<>("id"));

    nameTypeService.setCellValueFactory(new
PropertyValueFactory<>("name"));

    infoTypeService.setCellValueFactory(new
PropertyValueFactory<>("info"));

    termTypeService.setCellValueFactory(new
PropertyValueFactory<>("term"));

```

```

        countTypeService.setCellValueFactory(new
PropertyCellValueFactory<>("count"));

        costTypeService.setCellValueFactory(new
PropertyCellValueFactory<>("money"));
        typeTableView.setItems(typeServices);
    });
}

public void putData(Register register, long OrderID, String fullNameStaff) {
    this.register = register;
    this.OrderID = OrderID;
    this.fullNameStaff = fullNameStaff;
}
}

```

TakeOrders\Controller.java

```
package org.vitapasser.photocopypoint.Controller.TakeOrders;
```

```

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;

```

```
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import org.vitapasser.photocopypoint.MainApplication;
import org.vitapasser.photocopypoint.Model.Register;
import org.vitapasser.photocopypoint.Model.Ticket;
```

```
import java.io.IOException;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
```

```
public class Controller {
    ObservableList<Ticket> ticketsList = FXCollections.observableArrayList();

    private Register register;
    private String fullNameStaff = "";
```

```
@FXML
```

```
TableView<Ticket> listOfTicketsTableView;
```

```
public final Runnable changeValuesOnListOfTicketsTableView = () -> {
    try {
```

```
        while (!Thread.currentThread().isInterrupted()) {
```

```
            int
```

```
            index
```

```
=
```

```
listOfTicketsTableView.getSelectionModel().getFocusedIndex();
```

```
        List<Ticket> tickets = register.getAllNotMadeTickets(fullNameStaff);
```

```
        ticketsList.clear();
```

```

        ticketsList.addAll(tickets);
        listOfTicketsTableView.getSelectionModel().select(index);
        Thread.sleep(1000);
    }
} catch (Exception e) {
    throw new RuntimeException(e);
}
};

```

```

private final ExecutorService executor =
Executors.newSingleThreadExecutor(runnable -> {
    Thread t = new Thread(runnable);
    t.setDaemon(true);
    return t ;
});

```

@FXML

```
private TableColumn<Ticket, String> fullNameColumn;
```

@FXML

```
private TableColumn<Ticket, Long> idTicketColumn;
```

@FXML

```
private TableColumn<Ticket, String> nameTypeServiceColumn;
```

@FXML

```
private TableColumn<Ticket, String> phoneNumberColumn;
```

@FXML

```
private TextField fullNameTextField;
```

@FXML

private Button takeMakeOrderButton;

@FXML

```
protected void onTakeMakeOrderButtonClick(ActionEvent event) throws
IOException {
    if (ticketsList.isEmpty() || Objects.equals(fullNameStaff, "")) return;

    this.executor.shutdown();

    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("take-more-info-order-for-
make.fxml")));

    Scene scene = new Scene(FXMLLoader.load());
    org.vitapasser.photocopypoint.Controller.TakeMoreInfoForMake.Controller
controller = FXMLLoader.getController();
    controller.putData(register,
listOfTicketsTableView.getSelectionModel().getSelectedItem().getId(), fullNameStaff);
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.setTitle("Оглядач не виготовленого замовлення");
    stage.show();
}
```

@FXML

```
protected void onCancelButtonClick(ActionEvent event) throws IOException {
    FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
        MainApplication.class.getResource("order-management.fxml")));
    Scene scene = new Scene(FXMLLoader.load());
```

```

        org.vitapasser.photocopypoint.Controller.OrderManagement.Controller
controller = FXMLLoader.getController();
        controller.putData(register, fullNameStaff);
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.setTitle("Менеждер заомовлень");
        stage.show();
    }

    @FXML
    protected void onAuthorizationButtonClick(ActionEvent event) throws
IOException {
        authorization();
    }

    private void authorization()
    {
        fullNameStaff = fullNameTextField.getText();
        if(fullNameStaff.isEmpty()) return;

        changeTable();
        this.executor.submit(changeValuesOnListOfTicketsTableView);
        takeMakeOrderButton.setDisable(false);
    }

    private void changeTable() {
        List<Ticket> tickets = register.getAllNotMadeTickets(fullNameStaff);

        ticketsList.clear();
        ticketsList.addAll(tickets);
    }

```

```

    }

    public void initialize() {
        Platform.runLater(() -> {

            listOfTicketsTableView.setItems(ticketsList);
            ticketsList.clear();

            idTicketColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

            fullNameColumn.setCellValueFactory(new
PropertyCellValueFactory<>("fullName"));

            phoneNumberColumn.setCellValueFactory(new
PropertyCellValueFactory<>("phoneNumber"));

            namesTypeServiceColumn.setCellValueFactory(new
PropertyCellValueFactory<>("name"));

            fullNameTextField.setText(fullNameStaff);
            authorization();

        });
    }

    public void putData(Register register) {
        this.register = register;
    }

    public void putData(Register register, String fullNameStaff) {
        this.register = register;
        this.fullNameStaff = fullNameStaff;
    }
}

```



```
}
```

MainApplication.java

```
package org.vitapasser.photocopypoint;
```

```
import javafx.application.Application;
```

```
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Scene;
```

```
import javafx.stage.Stage;
```

```
import org.vitapasser.photocopypoint.Controller.OrderManagement.Controller;
```

```
import org.vitapasser.photocopypoint.Model.*;
```

```
import org.vitapasser.photocopypoint.Util.Mysql;
```

```
import java.sql.Connection;
```

```
import java.sql.SQLException;
```

```
import java.util.Objects;
```

```
public class MainApplication extends Application {
```

```
    Connection connection;
```

```
    @Override
```

```
    public void start(Stage stage){
```

```
        try{
```

```
            Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();
```

```
            System.out.println("Connection driver included!");
```

```
        } try {
```

```
connection = Mysql.getConnection();
```

```
TypeList typeList = new TypeList(connection);
```

```
PickUpStation pickUpStation = new PickUpStation(connection,  
    "Київ, вул. Хрещатик, 1");
```

```
TicketList ticketList = new TicketList(connection);
```

```
OrderList orderList = new OrderList(connection);
```

```
Register register = new Register(typeList, pickUpStation, ticketList,  
orderList);
```

```
FXMLLoader fxmlLoader = new  
FXMLLoader(Objects.requireNonNull(
```

```
    MainApplication.class.getResource("order-management.fxml"));
```

```
Scene scene = new Scene(fxmlLoader.load());
```

```
Controller controller = fxmlLoader.getController();
```

```
controller.putData(register);
```

```
stage.setTitle("Менеждер замовлень");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
} catch (Exception e) {
```

```
    System.out.println(e.getMessage());
```

```
}
```

```
}
```

```
catch(Exception ex){
```

```
    System.out.println("Connection failed...");
```

```
    System.out.println(ex.getMessage());
```

```
}
```

```

    }

    @Override
    public void stop() throws SQLException {
        connection.close();
    }

    public static void main(String[] args) {
        launch();
    }
}

Launcher.java
package org.vitapasser.photocopypoint;

import javafx.application.Application;

public class Launcher {
    public static void main(String[] args) {
        Application.launch(MainApplication.class, args);
    }
}

```

```

Module-info.java
module org.vitapasser.photocopypoint {
    requires javafx.controls;
    requires javafx.fxml;
    requires javafx.web;
}

```

```

requires org.controlsfx.controls;
requires com.dlsc.formsfx;
requires net.synedra.validatorfx;
requires org.kordamp.ikonli.javafx;
requires org.kordamp.bootstrapfx.core;
requires eu.hansolo.tilesfx;
requires com.almasb.fxgl.all;
requires mysql.connector.j;
requires java.sql;

```

```

opens org.vitapasser.photocopypoint to javafx.fxml;
exports org.vitapasser.photocopypoint;
exports org.vitapasser.photocopypoint.Controller;
opens org.vitapasser.photocopypoint.Controller to javafx.fxml;
exports org.vitapasser.photocopypoint.Model;
opens org.vitapasser.photocopypoint.Model to javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.CreatingOrder;
opens org.vitapasser.photocopypoint.Controller.CreatingOrder to javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.OrderManagement;
opens      org.vitapasser.photocopypoint.Controller.OrderManagement      to
javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.TakeMoreInfo;
opens org.vitapasser.photocopypoint.Controller.TakeMoreInfo to javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.TakeMoreInfoForMake;
opens      org.vitapasser.photocopypoint.Controller.TakeMoreInfoForMake      to
javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.DTO;
opens org.vitapasser.photocopypoint.Controller.DTO to javafx.fxml;
exports org.vitapasser.photocopypoint.Controller.TakeOrders;
opens org.vitapasser.photocopypoint.Controller.TakeOrders to javafx.fxml;

```

```
}
```

creating-order.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.geometry.Insets?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.ScrollPane?>
```

```
<?import javafx.scene.control.Separator?>
```

```
<?import javafx.scene.control.TableColumn?>
```

```
<?import javafx.scene.control.TableView?>
```

```
<?import javafx.scene.control.TextField?>
```

```
<?import javafx.scene.layout.BorderPane?>
```

```
<?import javafx.scene.layout.ColumnConstraints?>
```

```
<?import javafx.scene.layout.GridPane?>
```

```
<?import javafx.scene.layout.HBox?>
```

```
<?import javafx.scene.layout.RowConstraints?>
```

```
<?import javafx.scene.layout.VBox?>
```

```
<?import javafx.scene.text.Font?>
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
```

```
minWidth="-Infinity" prefHeight="800.0" prefWidth="1200.0"
```

```
xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
```

```
fx:controller="org.vitapasser.photocopypoint.Controller.CreatingOrder.Controller">
```

```
<right>
```

```
<BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">
```

```
<center>
```

```
<BorderPane BorderPane.alignment="CENTER">
```

```
<center>
```

```

    <VBox alignment="CENTER" prefHeight="323.0" prefWidth="200.0"
spacing="10.0">
    <children>
    <VBox alignment="CENTER">
    <children>
    <VBox prefHeight="31.0" prefWidth="200.0">
    <children>
    <Label fx:id="nameNameTypeService" text="Назва
послуги" />
    <TextField fx:id="nameTypeService"
onKeyTyped="#onChangeNameTypeServiceTextField" />
    </children>
    </VBox>
    <VBox alignment="CENTER_LEFT" prefHeight="39.0"
prefWidth="200.0">
    <children>
    <Label fx:id="countTypeServiceLabel" text="Кількість
послуги" />
    <TextField fx:id="countTypeService"
onKeyTyped="#onWriteCheckTextField" />
    </children>
    </VBox>
    <HBox alignment="CENTER">
    <VBox.margin>
    <Insets top="10.0" />
    </VBox.margin>
    <children>
    <Button fx:id="addTypeServiceButton"
contentDisplay="BOTTOM" mnemonicParsing="false"
onAction="#onAddTypeServiceButtonClick" text="Додати послугу" />

```

```

        </children>
        <opaqueInsets>
            <Insets />
        </opaqueInsets>
    </HBox>
</children>
</VBox>
<VBox fx:id="clientVBox" disable="true" layoutX="10.0"
layoutY="52.0" prefHeight="58.0" prefWidth="200.0">
    <children>
        <VBox layoutX="10.0" layoutY="10.0">
            <children>
                <Label text="Ім'я клієнта" />
                <TextField fx:id="nameClient"
onKeyTyped="#onWriteNamePhoneNumber" />
            </children>
        </VBox>
        <VBox layoutX="10.0" layoutY="10.0">
            <children>
                <Label fx:id="phoneNumberLabel" text="Номер телефону
клієнта" />
                <TextField fx:id="phoneNumberClient"
onKeyTyped="#onWriteNamePhoneNumber" />
            </children>
        </VBox>
    </children>
    <VBox layoutX="10.0" layoutY="459.0">
        <children>
            <Label text="Приблизний час вироблювання">

```

```

        <font>
            <Font size="21.0" />
        </font>
    </Label>
    <Label fx:id="resultTerm" text="00:00">
        <font>
            <Font size="24.0" />
        </font>
    </Label>
</children>
</VBox>
<VBox>
    <children>
        <Label text="Ціна замовлення">
            <font>
                <Font size="21.0" />
            </font>
        </Label>
        <Label fx:id="resultPrice" text="0.00 грн">
            <font>
                <Font size="24.0" />
            </font>
        </Label>
    </children>
</VBox>
</children>
<padding>
    <Insets top="10.0" />
</padding>
</VBox>

```



```

        </center>
        <padding>
            <Insets left="10.0" right="10.0" />
        </padding>
        <bottom>
            <VBox                alignment="CENTER"                spacing="10.0"
BorderPane.alignment="CENTER">
                <children>
                    <Button        fx:id="paymentButton"        alignment="CENTER"
disable="true"        mnemonicParsing="false"        onAction="#onPaymentButtonClick"
prefHeight="25.0" prefWidth="200.0" text="Оплата" />
                    <Button                mnemonicParsing="false"
onAction="#onCancelButtonClick"        prefHeight="25.0"        prefWidth="200.0"
text="Відмінити створення замовлення" />
                </children>
                <BorderPane.margin>
                    <Insets bottom="10.0" />
                </BorderPane.margin>
            </VBox>
        </bottom>
    </BorderPane>
</center>
<left>
    <Separator orientation="VERTICAL" BorderPane.alignment="CENTER"
/>

</left>
<BorderPane.margin>
    <Insets />
</BorderPane.margin></BorderPane>
</right>

```

```

<center>
  <GridPane BorderPane.alignment="CENTER">
    <columnConstraints>
      <ColumnConstraints          hgrow="SOMETIMES"          minWidth="10.0"
prefWidth="100.0" />
    </columnConstraints>
    <rowConstraints>
      <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
      <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
    </rowConstraints>
    <children>
      <BorderPane prefHeight="200.0" prefWidth="200.0">
        <top>
          <Label          text="Список          обираемых          услуг"
BorderPane.alignment="CENTER" />
        </top>
      </center>
      <ScrollPane fitToHeight="true" fitToWidth="true" prefHeight="200.0"
prefWidth="200.0" BorderPane.alignment="CENTER">
        <content>
          <TableView          fx:id="listOfSelectableServices"
onMouseClicked="#onGetNameTypeServiceTableView"          prefHeight="400.0"
prefWidth="736.0">
            <columnResizePolicy>
              <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
            </columnResizePolicy>
          </columns>

```

```

        <TableColumn    fx:id="idTypeColumn"    prefWidth="75.0"
text="Ід" />

        <TableColumn    fx:id="nameTypeColumn"    prefWidth="75.0"
text="Назва" />

        <TableColumn    fx:id="infoTypeColumn"    prefWidth="75.0"
text="Додаткова інформація" />

        <TableColumn    fx:id="termTypeColumn"    prefWidth="75.0"
text="Термін" />

        <TableColumn                                fx:id="countMoneyTypeColumn"
prefWidth="75.0" text="Ціна" />

        <TableColumn                                fx:id="currencyMoneyTypeColumn"
prefWidth="75.0" text="Валюта" />

        </columns>
    </TableView>
</content>
</ScrollPane>
</center></BorderPane>

<BorderPane    layoutX="10.0"    layoutY="10.0"    prefHeight="200.0"
prefWidth="200.0" GridPane.rowIndex="1">
    <top>
        <Label            text="Список            обраних            послуг"
BorderPane.alignment="CENTER" />
    </top>
    <center>
        <ScrollPane fitToHeight="true" fitToWidth="true" prefHeight="200.0"
prefWidth="200.0" BorderPane.alignment="CENTER">
            <content>
                <TableView    fx:id="listSelectedServices"    prefHeight="200.0"
prefWidth="200.0">
                    <columnResizePolicy>

```

```

        <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
        </columnResizePolicy>
        <columns>
            <TableColumn    fx:id="idTypeColumn1"    prefWidth="75.0"
text="Ід" />
            <TableColumn    fx:id="nameTypeColumn1"  prefWidth="75.0"
text="Назва" />
            <TableColumn    fx:id="infoTypeColumn1"  prefWidth="75.0"
text="Додаткова інформація" />
            <TableColumn    fx:id="termTypeColumn1"  prefWidth="75.0"
text="Термін" />
            <TableColumn    fx:id="countTypeColumn1" prefWidth="75.0"
text="Кількість" />
            <TableColumn                    fx:id="countMoneyTypeColumn1"
prefWidth="75.0" text="Ціна" />
            <TableColumn                    fx:id="currencyMoneyTypeColumn1"
prefWidth="75.0" text="Валюта" />
        </columns>
    </TableView>
</content>
</ScrollPane>
</center>
</BorderPane>
</children>
</GridPane>
</center>
</BorderPane>

```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.geometry.Insets?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.ScrollPane?>
```

```
<?import javafx.scene.control.Separator?>
```

```
<?import javafx.scene.control.TableColumn?>
```

```
<?import javafx.scene.control.TableView?>
```

```
<?import javafx.scene.layout.BorderPane?>
```

```
<?import javafx.scene.layout.VBox?>
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity"                prefHeight="800.0"                prefWidth="1200.0"
xmlns="http://javafx.com/javafx/21"                xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.vitapasser.photocopypoint.Controller.OrderManagement.Controller">
```

```
<right>
```

```
<BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">
```

```
<left>
```

```
<Separator
```

```
orientation="VERTICAL"
```

```
BorderPane.alignment="CENTER" />
```

```
</left>
```

```
<BorderPane.margin>
```

```
<Insets />
```

```
</BorderPane.margin>
```

```
<center>
```

```
<VBox alignment="CENTER" prefHeight="200.0" prefWidth="100.0"
spacing="20.0">
```

```
<children>
```

```

        <Button      alignment="CENTER"      mnemonicParsing="false"
onAction="#onGetMoreInfoOrderButtonClick"  prefHeight="43.0"  prefWidth="200.0"
text="Отримати додаткову інформацію щодо квитка" wrapText="true" />

        <Button      alignment="CENTER"      mnemonicParsing="false"
onAction="#onStartCreateOrderButtonClick"  prefHeight="25.0"  prefWidth="200.0"
text="Створити нове замовлення" />

        <Button      alignment="CENTER"      mnemonicParsing="false"
onAction="#onGetMoreNotMadeOrdersButtonClick"      prefHeight="25.0"
prefWidth="200.0" text="Виконати нове замовлення" />

    </children>

</VBox>

</center></BorderPane>

</right>

<center>

    <BorderPane      prefHeight="200.0"      prefWidth="200.0"
BorderPane.alignment="CENTER">

        <top>

            <Label text="Готові замовлення" BorderPane.alignment="CENTER" />

        </top>

        <center>

            <ScrollPane fitToHeight="true" fitToWidth="true">

                <content>

                    <TableView  fx:id="listOfTicketsTableView"  prefHeight="200.0"
prefWidth="200.0">

                        <columns>

                            <TableColumn  fx:id="idTicketColumn"  prefWidth="75.0"
text="Ід квитка" />

                            <TableColumn  fx:id="fullNameColumn"  prefWidth="75.0"
text="ПІБ" />

```

```

        <TableColumn fx:id="phoneNumberColumn" prefWidth="75.0"
text="Телефон" />
        <TableColumn fx:id="namesTypeServiceColumn"
prefWidth="75.0" text="Назви послуг" />
    </columns>
    <columnResizePolicy>
        <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
    </columnResizePolicy>
</TableView>
</content>
</ScrollPane>
</center>
</BorderPane>
</center>
</BorderPane>

```

payment.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.VBox?>

```

```
<?import javafx.scene.text.Font?>
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity"          prefHeight="800.0"          prefWidth="1200.0"
xmlns="http://javafx.com/javafx/21"          xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.vitapasser.photocopypoint.Controller.PaymentController">
```

```
<right>
```

```
<BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">
```

```
<center>
```

```
<VBox alignment="CENTER" spacing="30.0">
```

```
<children>
```

```
<VBox spacing="10.0">
```

```
<children>
```

```
<VBox alignment="CENTER_LEFT">
```

```
<children>
```

```
<Label fx:id="payLabel" text="До сплати">
```

```
<font>
```

```
<Font size="21.0" />
```

```
</font>
```

```
</Label>
```

```
<Label fx:id="resultPrice" text="0.00 грн">
```

```
<font>
```

```
<Font size="24.0" />
```

```
</font>
```

```
</Label>
```

```
</children>
```

```
</VBox>
```

```
<VBox          alignment="CENTER_LEFT"          prefHeight="39.0"
prefWidth="200.0" spacing="5.0">
```

```
<children>
```



```

        <Label text="Внесена оплата клієнтом" />
        <TextField fx:id="countClientPayTextField" />
    </children>
</VBox>
    <Button      fx:id="acceptButton"      mnemonicParsing="false"
onAction="#onPaymentButtonClick" text="Пітвердити оплату" />
    <Button      fx:id="cancelButton"      mnemonicParsing="false"
onAction="#onCancelButtonClick" text="Відменити оплату" />
</children>
</VBox>
    <VBox      fx:id="oddPayVBox"      disable="true"      layoutX="10.0"
layoutY="92.0" spacing="10.0">
    <children>
        <VBox      alignment="CENTER_LEFT"      prefHeight="10.0"
prefWidth="193.0">
    <children>
        <Label text="Решта">
            <font>
                <Font size="21.0" />
            </font>
        </Label>
        <Label fx:id="resultOddMoney" text="0.00 грн">
            <font>
                <Font size="24.0" />
            </font>
        </Label>
    </children>
    </VBox>
    <Button      mnemonicParsing="false"
onAction="#onOrderManagementButtonClick" text="Повернутися до меню" />

```

```

        </children>
    </VBox>
</children>
<padding>
    <Insets left="10.0" right="10.0" />
</padding>
</VBox>
</center>
<left>
    <Separator orientation="VERTICAL" BorderPane.alignment="CENTER"
/>

</left>
</BorderPane>
</right>
<center>
    <BorderPane BorderPane.alignment="CENTER">
        <center>
            <ScrollPane fitToHeight="true" fitToWidth="true" prefHeight="200.0"
prefWidth="200.0" BorderPane.alignment="CENTER">
                <content>
                    <TableView fx:id="listSelectedServices" prefHeight="200.0"
prefWidth="200.0">
                        <columnResizePolicy>
                            <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
                        </columnResizePolicy>
                        <columns>
                            <TableColumn fx:id="idTypeColumn1" prefWidth="75.0"
text="Ід" />

```

```

        <TableColumn fx:id="nameTypeColumn1" prefWidth="75.0"
text="Назва" />
        <TableColumn fx:id="infoTypeColumn1" prefWidth="75.0"
text="Додаткова інформація" />
        <TableColumn fx:id="termTypeColumn1" prefWidth="75.0"
text="Термін" />
        <TableColumn fx:id="countTypeColumn1" prefWidth="75.0"
text="Кількість" />
        <TableColumn fx:id="countMoneyTypeColumn1"
prefWidth="75.0" text="Ціна" />
        <TableColumn fx:id="currencyMoneyTypeColumn1"
prefWidth="75.0" text="Валюта" />
    </columns>
</TableView>
</content>
</ScrollPane>
</center>
<top>
    <Label text="Послуги у замовлені" BorderPane.alignment="CENTER" />
</top>
</BorderPane>
</center>
</BorderPane>

```

take-more-info-order.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>

```

```

<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>

```

```

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity"          prefHeight="800.0"          prefWidth="1200.0"
xmlns="http://javafx.com/javafx/21"          xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.vitapasser.photocopypoint.Controller.TakeMoreInfo.Controller">

```

```

<right>

```

```

    <BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">

```

```

        <center>

```

```

            <VBox alignment="CENTER" spacing="30.0">

```

```

                <children>

```

```

                    <VBox          fx:id="oddPayVBox"          alignment="TOP_CENTER"
layoutX="10.0" layoutY="92.0" spacing="10.0">

```

```

                        <children>

```

```

                            <Button                                mnemonicParsing="false"
onAction="#onSetGiveTheOrderButtonClick"          prefWidth="200.0"          text="Віддати
заовлення" />

```

```

                            <Button                                mnemonicParsing="false"
onAction="#onOrderManagementButtonClick" prefWidth="200.0" text="Повернутися до
меню" />

```

```

                        </children>

```

```

        </VBox>
    </children>
    <padding>
        <Insets left="10.0" right="10.0" />
    </padding>
</VBox>
</center>
<left>
    <Separator orientation="VERTICAL" BorderPane.alignment="CENTER"
/>

</left>
</BorderPane>
</right>
<center>
    <GridPane BorderPane.alignment="CENTER">
        <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />
            <RowConstraints          percentHeight="50.0"          valignment="CENTER"
vgrow="SOMETIMES" />
        </rowConstraints>
        <children>
            <BorderPane>
                <center>
                    <ScrollPane fitToHeight="true" fitToWidth="true">
                        <content>
                            <ListView fx:id="listInfo" prefHeight="200.0" prefWidth="200.0"
/>

```

```

        </content>
    </ScrollPane>
</center>
<top>
    <Label          text="Інформація          о          замовлені"
BorderPane.alignment="CENTER" />
    </top>
</BorderPane>
    <BorderPane          prefHeight="200.0"          prefWidth="200.0"
GridPane.rowIndex="1">
    <top>
        <Label      text="Інформація      о      послугах      у      замовлені"
BorderPane.alignment="CENTER" />
        </top>
        <center>
            <ScrollPane          fitToHeight="true"          fitToWidth="true"
BorderPane.alignment="CENTER">
                <content>
                    <TableView          fx:id="typeTableView"          prefHeight="200.0"
prefWidth="200.0">
                        <columns>
                            <TableColumn      fx:id="nameTypeService"      prefWidth="75.0"
text="Назва" />
                            <TableColumn      fx:id="idTypeService"          prefWidth="75.0"
text="Ід" />
                            <TableColumn      fx:id="infoTypeService"          prefWidth="75.0"
text="Додаткова інформація" />
                            <TableColumn      fx:id="termTypeService"          prefWidth="75.0"
text="Приблизний термін створення" />

```

```

        <TableColumn fx:id="costTypeService" prefWidth="75.0"
text="Ціна" />
        <TableColumn fx:id="countTypeService" prefWidth="75.0"
text="Кількість" />
    </columns>
    <columnResizePolicy>
        <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
    </columnResizePolicy>
</TableView>
</content>
</ScrollPane>
</center>
</BorderPane>
</children>
</GridPane>
</center>
</BorderPane>

```

take-more-info-order-for-make.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>

```

```

<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>

```

```

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity"                prefHeight="800.0"                prefWidth="1200.0"
xmlns="http://javafx.com/javafx/21"                xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.vitapasser.photocopypoint.Controller.TakeMoreInfoForMake.Controlle
r">

```

```

<right>

```

```

<BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">

```

```

<center>

```

```

<VBox alignment="CENTER" spacing="30.0">

```

```

<children>

```

```

<VBox fx:id="oddPayVBox" alignment="TOP_CENTER"
layoutX="10.0" layoutY="92.0" spacing="10.0">

```

```

<children>

```

```

<Button mnemonicParsing="false"
onAction="#onSetGiveTheOrderButtonClick" prefWidth="200.0" text="Замовлення
виконано" />

```

```

<Button mnemonicParsing="false"
onAction="#onTakeOrderButtonClick" prefWidth="200.0" text="Повернутися до меню"
/>

```

```

</children>

```

```

</VBox>

```

```

</children>

```

```

<padding>

```

```

<Insets left="10.0" right="10.0" />

```



```

        </padding>
    </VBox>
</center>
<left>
    <Separator orientation="VERTICAL" BorderPane.alignment="CENTER"
/>

    </left>
</BorderPane>
</right>
<center>
    <GridPane BorderPane.alignment="CENTER">
        <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />
            <RowConstraints      percentHeight="50.0"      valignment="CENTER"
vgrow="SOMETIMES" />
        </rowConstraints>
        <children>
            <BorderPane>
                <center>
                    <ScrollPane fitToHeight="true" fitToWidth="true">
                        <content>
                            <ListView fx:id="listInfo" prefHeight="200.0" prefWidth="200.0"
/>

                        </content>
                    </ScrollPane>
                </center>
                <top>

```

```

        <Label          text="Інформація          о          замовлені"
BorderPane.alignment="CENTER" />
    </top>
</BorderPane>

    <BorderPane          prefHeight="200.0"          prefWidth="200.0"
GridPane.rowIndex="1">
    <top>
        <Label      text="Інформація      о      послугах      у      замовлені"
BorderPane.alignment="CENTER" />
    </top>
    <center>
        <ScrollPane          fitToHeight="true"          fitToWidth="true"
BorderPane.alignment="CENTER">
    <content>
        <TableView          fx:id="typeTableView"          prefHeight="200.0"
prefWidth="200.0">
    <columns>
        <TableColumn      fx:id="nameTypeService"      prefWidth="75.0"
text="Назва" />
        <TableColumn      fx:id="idTypeService"          prefWidth="75.0"
text="Ід" />
        <TableColumn      fx:id="infoTypeService"        prefWidth="75.0"
text="Додаткова інформація" />
        <TableColumn      fx:id="termTypeService"        prefWidth="75.0"
text="Приблизний термін створення" />
        <TableColumn      fx:id="costTypeService"        prefWidth="75.0"
text="Ціна" />
        <TableColumn      fx:id="countTypeService"       prefWidth="75.0"
text="Кількість" />
    </columns>

```

```

        <columnResizePolicy>
            <TableView
fx:constant="CONSTRAINED_RESIZE_POLICY" />
        </columnResizePolicy>
    </TableView>
</content>
</ScrollPane>
</center>
</BorderPane>
</children>
</GridPane>
</center>
</BorderPane>

```

take-orders.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.VBox?>

```

```

    <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity"                prefHeight="800.0"                prefWidth="1200.0"

```

```

xmlns="http://javafx.com/javafx/21"                xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.vitapasser.photocopypoint.Controller.TakeOrders.Controller">
    <right>
        <BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">
            <left>
                <Separator orientation="VERTICAL" BorderPane.alignment="CENTER"
/>

            </left>
            <BorderPane.margin>
                <Insets />
            </BorderPane.margin>
            <center>
                <VBox alignment="CENTER" prefHeight="200.0" prefWidth="100.0"
spacing="20.0">
                    <children>
                        <VBox alignment="TOP_CENTER" fillWidth="false"
prefHeight="42.0" prefWidth="368.0">
                            <children>
                                <VBox spacing="5.0">
                                    <children>
                                        <Label text="ФІО виконавця" />
                                        <TextField fx:id="fullNameTextField" alignment="CENTER"
prefWidth="200.0" />
                                        <Button mnemonicParsing="false"
onAction="#onAuthorizationButtonClick" prefWidth="200.0" text="Авторизуватись" />
                                    </children>
                                </VBox>
                            </children>
                        <VBox.margin>
                            <Insets />

```

```

        </VBox.margin>
    </VBox>
    <Button    fx:id="takeMakeOrderButton"    alignment="CENTER"
disable="true"    mnemonicParsing="false"    onAction="#onTakeMakeOrderButtonClick"
prefHeight="25.0" prefWidth="200.0" text="Взяти на виконання замовлення" />
    <Button    alignment="CENTER"    mnemonicParsing="false"
onAction="#onCancelButtonClick"    prefHeight="25.0"    prefWidth="200.0"
text="Повернутись назад" />
    </children>
    <BorderPane.margin>
    <Insets left="10.0" right="10.0" />
    </BorderPane.margin>
</VBox>
</center></BorderPane>
</right>
<center>
    <BorderPane    prefHeight="200.0"    prefWidth="200.0"
BorderPane.alignment="CENTER">
    <top>
    <Label    text="Не    виконині    замовлення"
BorderPane.alignment="CENTER" />
    </top>
    <center>
    <ScrollPane fitToHeight="true" fitToWidth="true">
    <content>
    <TableView    fx:id="listOfTicketsTableView"    prefHeight="200.0"
prefWidth="200.0">
    <columns>
    <TableColumn    fx:id="idTicketColumn"    prefWidth="75.0"
text="Ід квитка" />

```

```

        <TableColumn fx:id="fullNameColumn" prefWidth="75.0"
text="ПІБ" />

        <TableColumn fx:id="phoneNumberColumn" prefWidth="75.0"
text="Телефон" />

        <TableColumn fx:id="namesTypeServiceColumn"
prefWidth="75.0" text="Назви послуг" />
    </columns>
    <columnResizePolicy>
        <TableView fx:constant="CONSTRAINED_RESIZE_POLICY"
/>

    </columnResizePolicy>
</TableView>
</content>
</ScrollPane>
</center>
</BorderPane>
</center>
</BorderPane>

```

ДОДАТОК В СТРУКТУРА ФАЙЛІВ КОДУ ПРОГРАМИ

На рисунку .1 та .2 зображена структура файлів код і папок проекту.

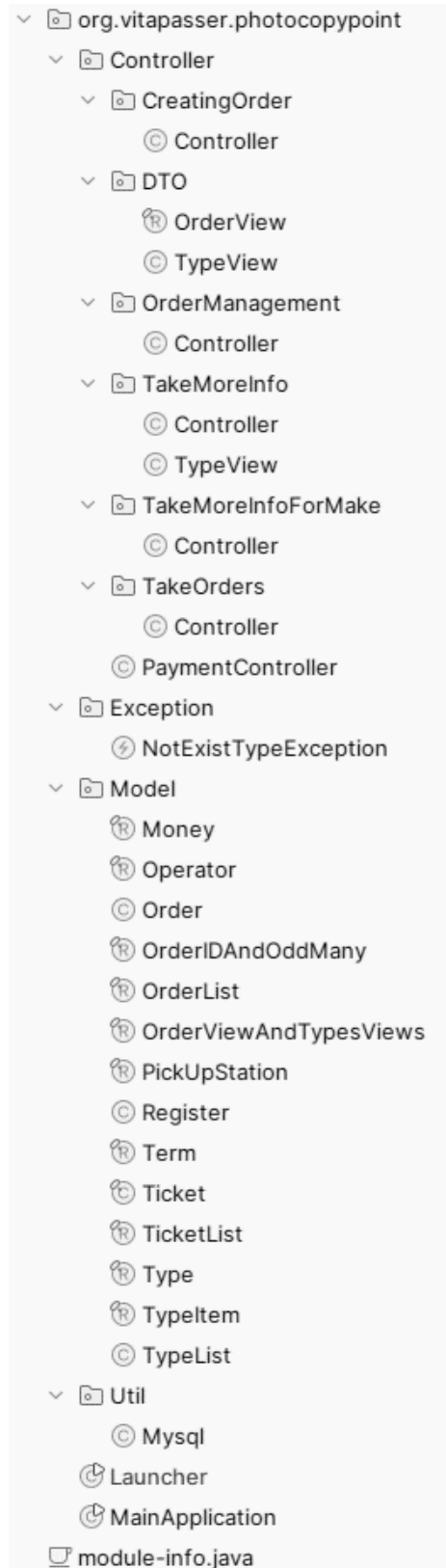


Рисунок 7.1 Структура файлів і папок проекту.

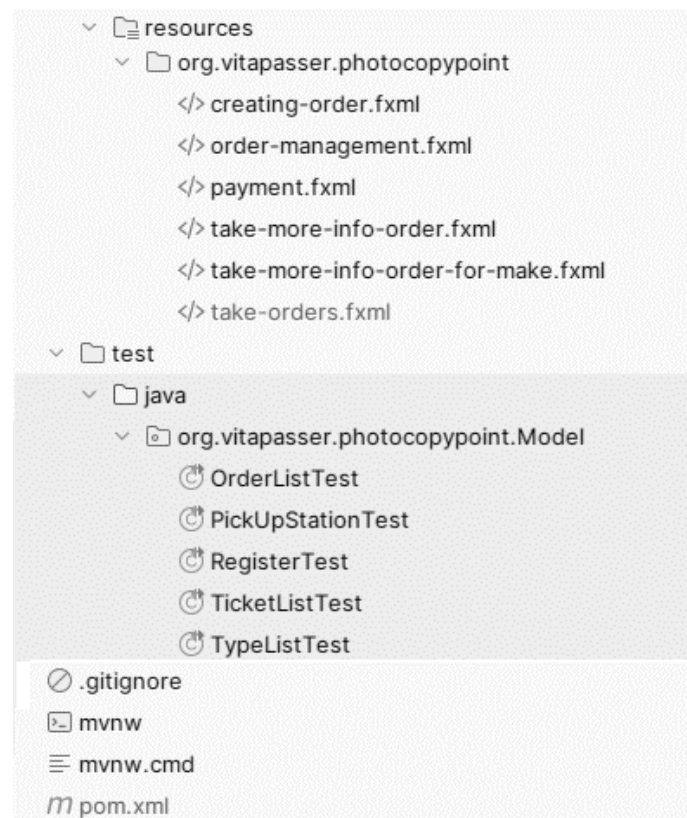


Рисунок 7.2 Структура файлів розмітки користувацького графічного інтерфейсу, тестів, розмітки конфігурації maven і папок проекту.

ДОДАТОК Г КОД КОНФІГУРАЦІЇ ДЛЯ MAVEN

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>org.vitapasser</groupId>
```

```
    <artifactId>PhotocopyPoint</artifactId>
```

```
    <version>1.0-SNAPSHOT</version>
```

```
    <name>PhotocopyPoint</name>
```

```
    <properties>
```

```
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
    <junit.version>5.10.0</junit.version> </properties>
```

```
    <dependencies>
```

```
        <dependency>
```

```
            <groupId>mysql</groupId>
```

```
            <artifactId>mysql-connector-java</artifactId>
```

```
            <version>8.0.33</version>
```

```
        </dependency>
```

```
        <dependency>
```

```
            <groupId>org.openjfx</groupId>
```

```
            <artifactId>javafx-controls</artifactId>
```

```
            <version>21</version>
```

```
        </dependency>
```

```

<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-fxml</artifactId>
  <version>21</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-web</artifactId>
  <version>21</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-swing</artifactId>
  <version>21</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-media</artifactId>
  <version>21</version>
</dependency><dependency>
  <groupId>org.controlsfx</groupId>
  <artifactId>controlsfx</artifactId>
  <version>11.1.2</version>
</dependency><dependency>
  <groupId>com.dlsc.formsfx</groupId>
  <artifactId>formsfx-core</artifactId>
  <version>11.6.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>

```

```

    <artifactId>*</artifactId>
  </exclusion>
</exclusions>
</dependency><dependency>
  <groupId>net.synedra</groupId>
  <artifactId>validatorfx</artifactId>
  <version>0.4.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency><dependency>
  <groupId>org.kordamp.ikonli</groupId>
  <artifactId>ikonli-javafx</artifactId>
  <version>12.3.1</version>
</dependency><dependency>
  <groupId>org.kordamp.bootstrapfx</groupId>
  <artifactId>bootstrapfx-core</artifactId>
  <version>0.4.0</version>
</dependency><dependency>
  <groupId>eu.hansolo</groupId>
  <artifactId>tilesfx</artifactId>
  <version>11.48</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>

```

```

    </exclusions>
</dependency><dependency>
  <groupId>com.github.almasb</groupId>
  <artifactId>fxgl</artifactId>
  <version>17.3</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency> </dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>

```

```

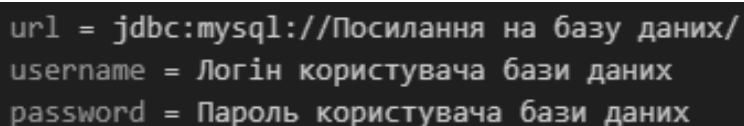
<version>3.11.0</version>
<configuration>
  <source>21</source>
  <target>21</target>
</configuration>
</plugin>
<plugin>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-maven-plugin</artifactId>
  <version>0.0.8</version>
  <executions>
    <execution>
      <!-- Default configuration for running with: mvn clean javafx:run -->
      <id>default-cli</id>
      <configuration>

<mainClass>org.vitapasser.photocopypoint/org.vitapasser.photocopypoint.MainApplication
n</mainClass>
      <launcher>app</launcher>
      <jlinkZipName>app</jlinkZipName>
      <jlinkImageName>app</jlinkImageName>
      <noManPages>true</noManPages>
      <stripDebug>true</stripDebug>
      <noHeaderFiles>true</noHeaderFiles>
    </execution>
  </executions>
</plugin>
</plugins>
</build></project>

```

ДОДАТОК Д ІНСТРУКЦІЯ КОРИСТУВАЧА

Після встановлення програмного забезпечення потрібно налаштувати зв'язок до бази даних. Для цього потрібно зайти у папку з програмою, яка по замовченню для windows встановлюється по шляху «C:\Program Files (x86)\Photocopy point app». Далі зайти у папку «Configuration» і через текстовий редактор відкрити файл «database» (або з розширенням «database.properties»). На рисунку 8.1 зображен зміст файлу з заглушками, які потрібно замінити відповідними значеннями. У url заміщається «Посилання на базу даних» на адресу до бази даних. У «Логін користувача бази даних» заміщається пароль до користувача для бази даних. У «Пароль користувача бази даних» заміщається паролем користувача бази даних.



```
url = jdbc:mysql://Посилання на базу даних/  
username = Логін користувача бази даних  
password = Пароль користувача бази даних
```

Рисунок 8.1 – Конфігураційний файл програми для підключення до бази даних

Після налаштування програми можливо її запустити через ярлик з назвою «Photocopy point app».

Після запуску програми вас буде зустрічати вікно менеджера замовлень, рисунок 8.2, з наступними елементами: З таблицею готових замовлень, які потрібно віддати замовнику, кнопки для отримання додаткової інформації щодо вибраного замовлення, створення нового замовлення, виконання нового замовлення. Це вікно призначена касиру для отримання повідомлення щодо готових замовлень та оператору з касиром для переходу до створення нового замовлення або виконання нового замовлення відповідно.

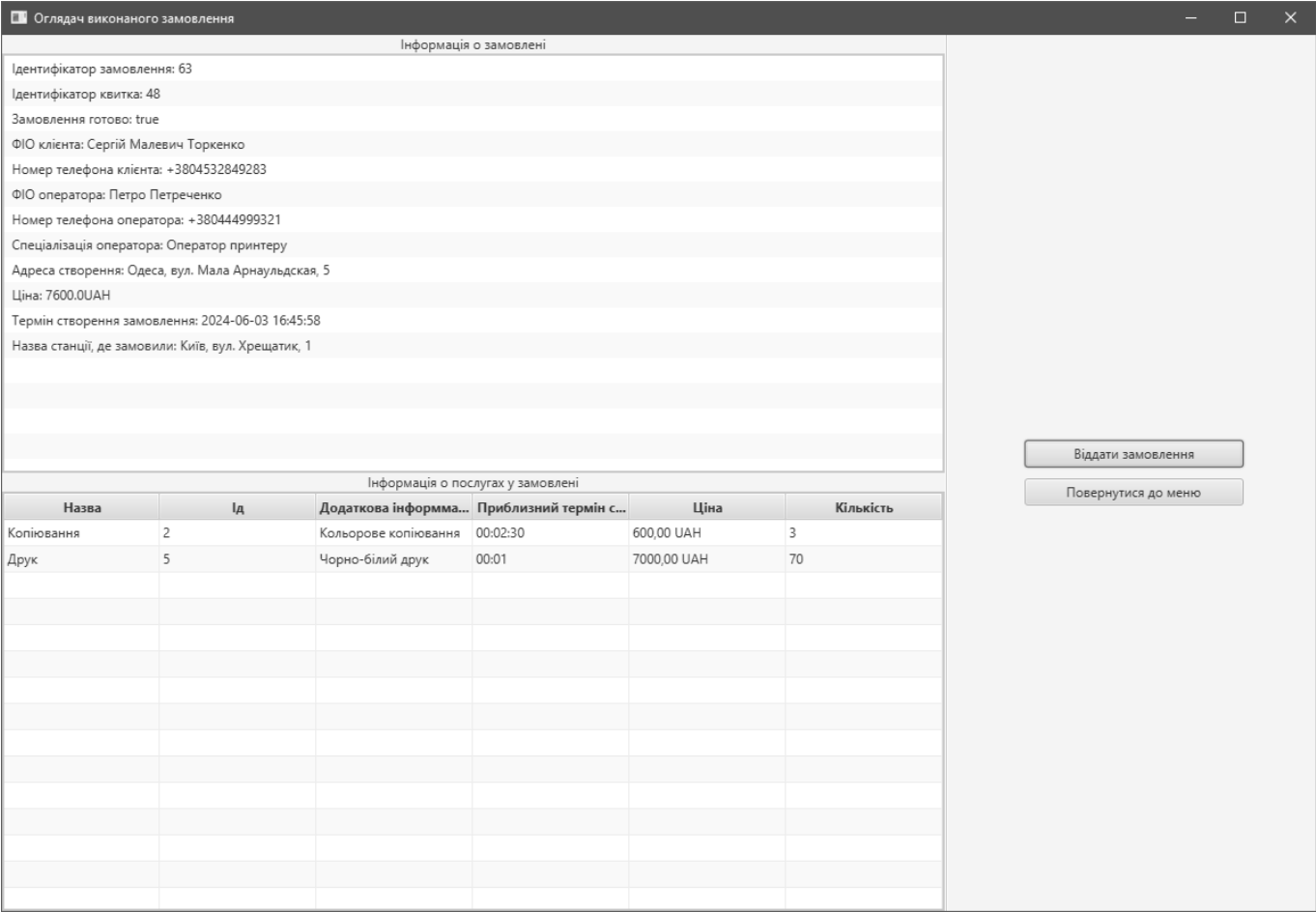


Рисунок 8.3 – Вікно оглядача виконаного замовлення

Виконання існуючого замовлення

Не виконані замовлення

Ід квитка	ПІБ	Телефон	Назви послуг
No content in table			

ФІО виконавця

Авторизуватись

Взяти на виконання замовлення

Повернутись назад

Рисунок 8.5 – Вікно виконання існуючого замовлення

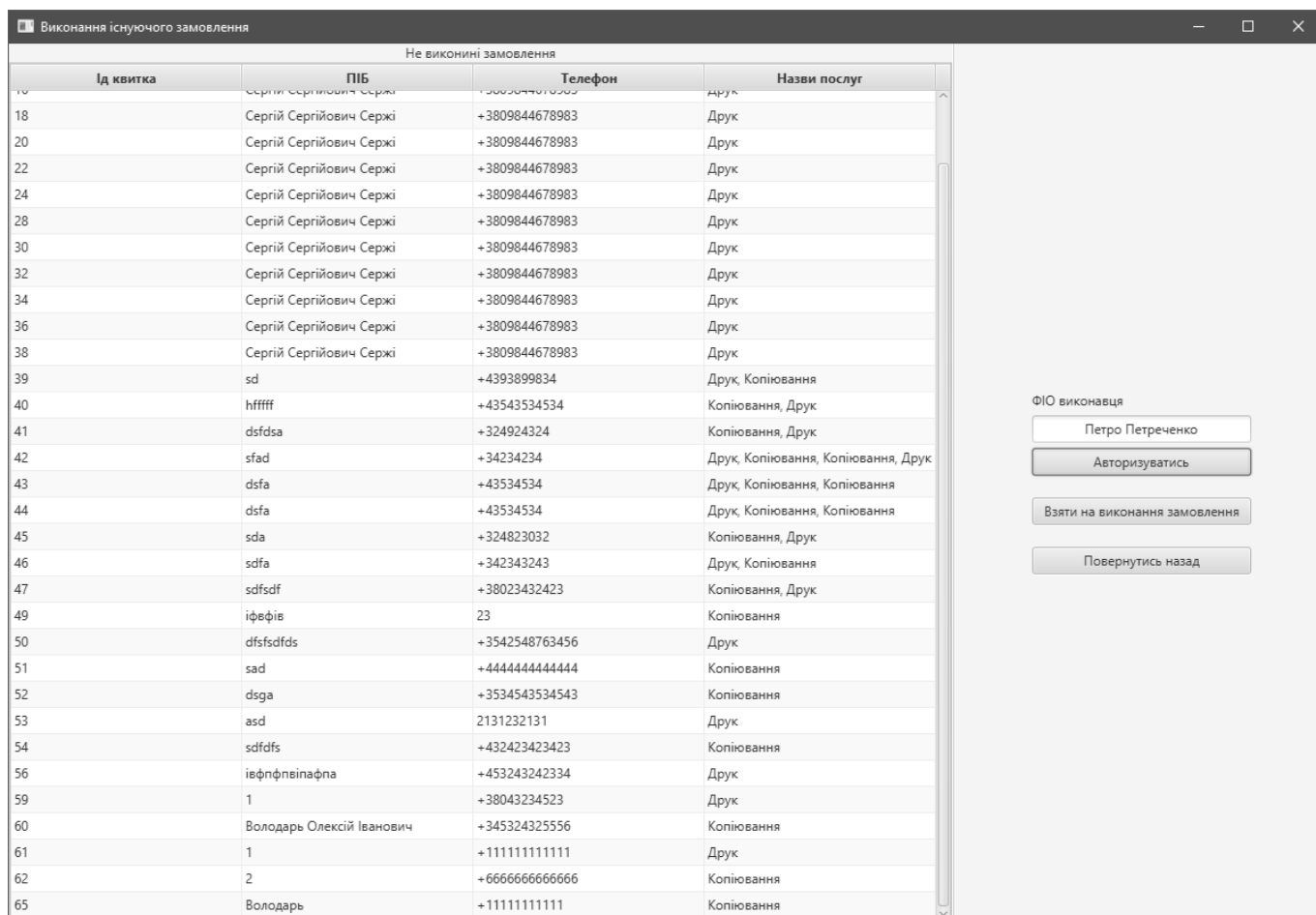


Рисунок 8.6 – Вікно виконання існуючого замовлення після авторизації

Оглядач не виготовленого замовлення

Інформація о замовленні

Ідентифікатор замовлення: 77

Ідентифікатор квитка: 60

Замовлення готово: false

ФІО клієнта: Володарь Олексій Іванович

Номер телефона клієнта: +345324325556

ФІО оператора: Петро Петrenchенко

Номер телефона оператора: +380444999321

Спеціалізація оператора: Оператор принтеру

Адреса створення: Одеса, вул. Мала Арнаульдская, 5

Ціна: 600.0UAH

Термін створення замовлення: 2024-06-09 10:41:23

Назва станції, де замовили: Київ, вул. Хрещатик, 1

Інформація о послугах у замовленні

Назва	Id	Додаткова інформа...	Приблизний термін с...	Ціна	Кількість
Копіювання	2	Кольорове копіювання	00:02:30	600,00 UAH	3

Замовлення виконано

Повернутися до меню

Рисунок 8.7 – Вікно оглядача не виготовленого замовлення

Створення нового замовлення

Список обираємих послуг

Ід	Назва	Додаткова інформація	Термін	Ціна	Валюта
5	Друк	Чорно-білий друк	00:01	100.0	UAH
2	Копіювання	Кольорове копіювання	00:02:30	200.0	UAH

Список обраних послуг

Ід	Назва	Додаткова інфор...	Термін	Кількість	Ціна	Валюта
No content in table						

Назва послуги

Кількість послуги

Додати послугу

Ім'я клієнта

Номер телефону клієнта

Приблизний час вироблювання
 00:00

Ціна замовлення
 0.00 грн

Оплата

Відмінити створення замовлення

Рисунок 8.9 –Створення нового замовлення

У вікні «Створення нового замовлення» відображується таблиця «Список обраних послуг» для послуг, які були знайдені через поле «Назва послуг». Для доданих послуг є «Список обраних послуг», у якого додані послуги та їх кількість через поле «Кількість послуг» та нажаті кнопки «Додати послугу». Після додаванні хоча б одної послуги до замовлення можливо написати ПІБ клієнта та його номер телефона, рисунок 8.10. Приблизний час вироблення та сукупна ціна замовлення відображає відповідні значення для замовлення в цілому. Після додавання усієї потрібної інформації можливо нажати на кнопку «Оплата», рисунок 8.11, та перейти у вікно оплати, рисунок 8.12.

[illegible]

Рисунок 8.10 – Вікно створення нового замовлення після додавання послуги
«Копіювання»

[illegible]

Рисунок 8.11 – Вікно створення нового замовлення після додавання послуги «Копіювання» та надання ПІБ та номер телефону клієнта

[illegible]

Рисунок 8.12 – Вікно «Оплати замовлення»

У вікні «Оплата замовлення» відображається список обраних послуг та кількість грошей необхідних для сплати замовлення клієнтом. У поле «Внесена оплата клієнтом» записується кількість грошей наданих клієнтом і якщо цих грошей не достатньо то замовлення не буде зафіксовано і з'явиться повідомлення о потреби надання більшої суми клієнтом. Якщо суми було достатньо тоді замовлення за фіксується у базі даних і буде відображатись у вікні «не виготовлені замовлення», а вікно перейде у вікно «Менеджер замовлень».