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

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 привиду 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 бумаги та 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 будете 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 послідуюі положення.

2.1.3 Опис користувачів

Опис користувачів наддасть 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 середа.

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 бухгалтеру, оператору, 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 іншій 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 скора зміна системного оснащення.
Експлуатаційний	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 привиду 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 послугую 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 надаваємої 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 надаваємої 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 надаваємої 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.

5а2. Касир може спробувати переговорити щодо ціни або запропонувати альтернативні варіанти послуги з іншою вартістю.

5а3. Якщо клієнт все одно відмовляється, замовлення відміняють.

6а. Касир вносить неправильну інформацію:

6а1. Касир вводить інформацію у систему. Система виводить помилку вводу.

6а2. Касир просить Клієнта надати ще раз дані. Касир вводить їх у систему.

6б3. Якщо система знову виводить помилку, замовлення відміняється, 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 послуг.

Мінімальна гарантія: Клієнт повертається за послугою до каси.

Гарантія успіху: Клієнт отримує замовлену послугу.

Тригер: Замовлена послуга створена.

Основний сценарій:

1. Система повідомляє клієнта о кінцевим статусі послуги. Якщо замовлення готово, то повідомляє о його завершеності та потребує підійти до каси.

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.

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 послугую. 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 надаваємої 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 надаваємої послуги. Касир говорить клієнту отриману з системи вартість послуги. Клієнт згоден з вартістю.	Order, PriceList, Money, Register
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 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.	

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

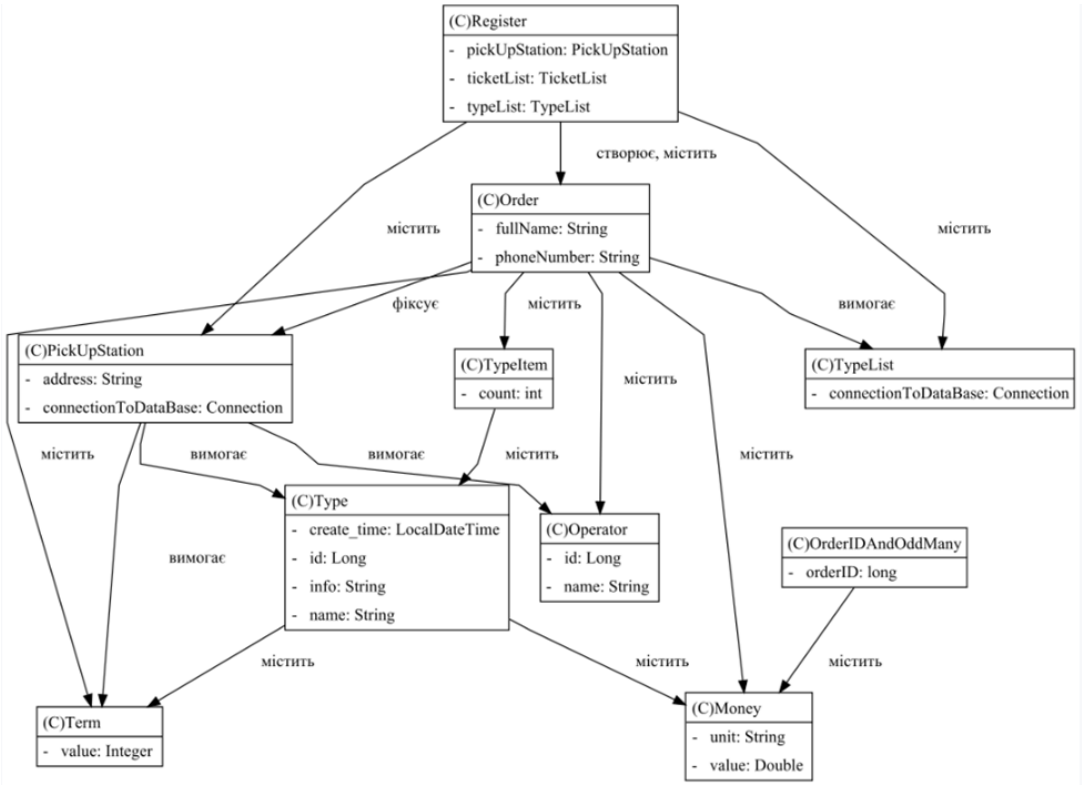

Рисунок 2.1 – Діаграма концептуальних класів прецеденту «Замовлення послуги»

3.3 загальна діаграма концептуальних класів:

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

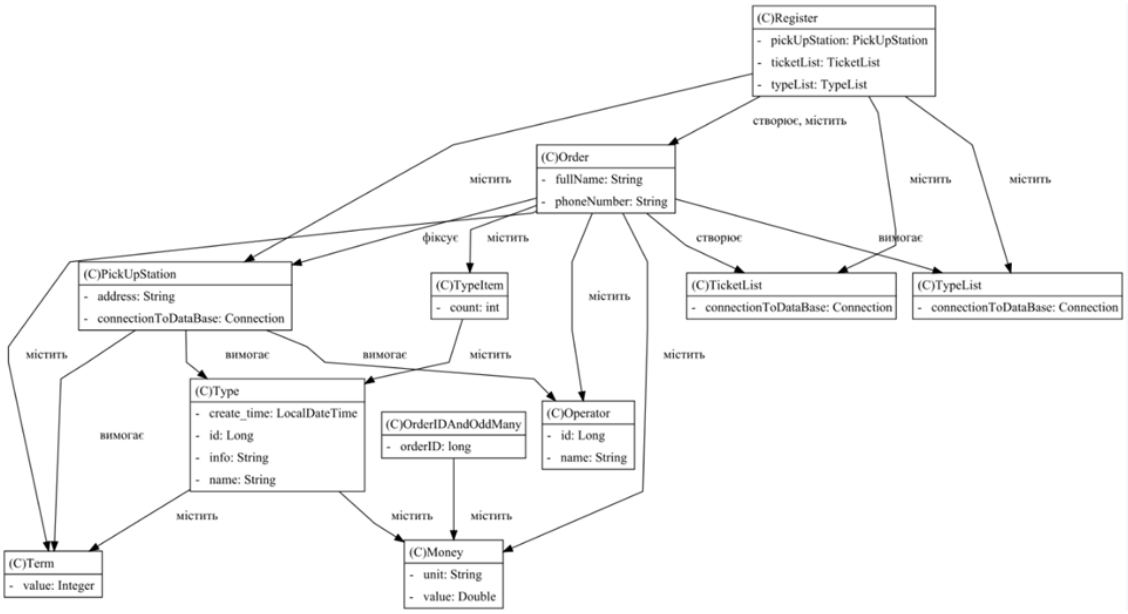

Рисунок 2.2 - Діаграма концептуальних класів прецеденту «Замовлення послуги» і «Отримання послуги», тобто загальна

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.

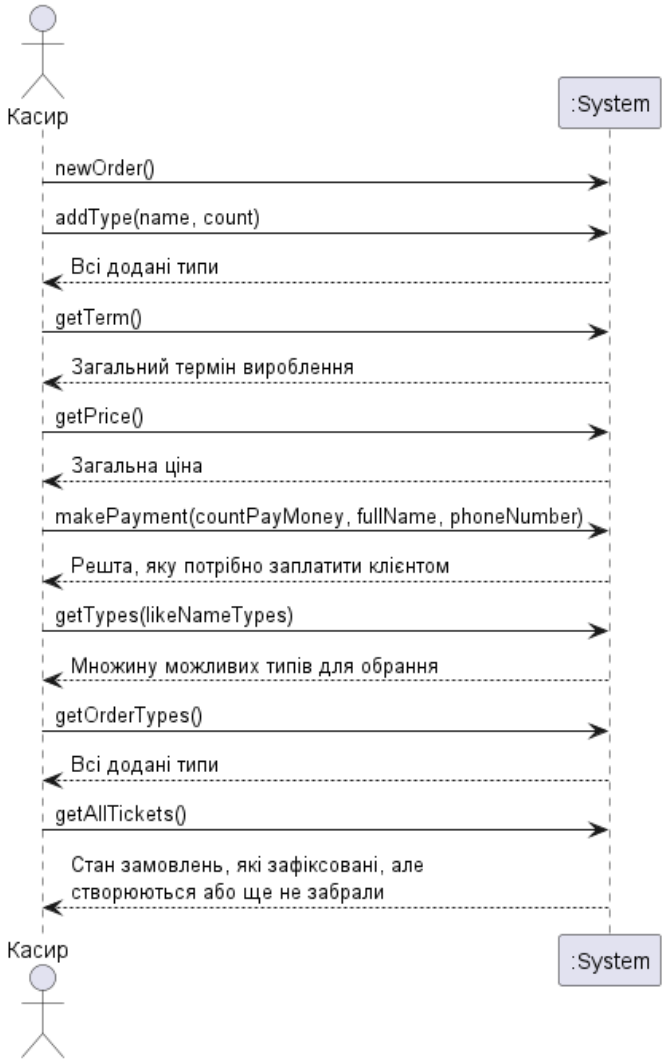

Рисунок 3.1 - Системні операції на підставі ВВ «Замовлення послуги»

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 послугую. Касир створює нове замовлення у системі. Система фіксує касира, який наддає замовлення.

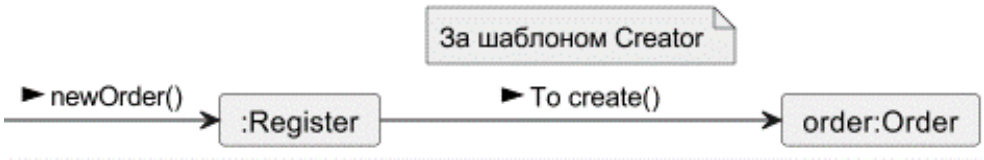

Рисунок 4.1 - Проектне рішення newOrder

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.

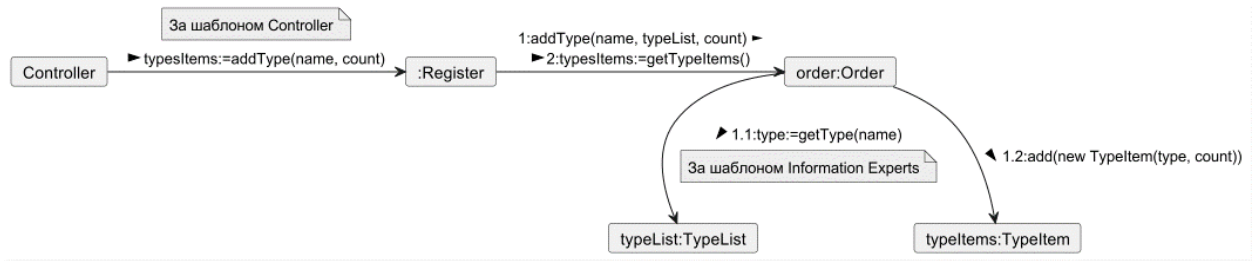

Рисунок 4.2 - Проектне рішення addType

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.

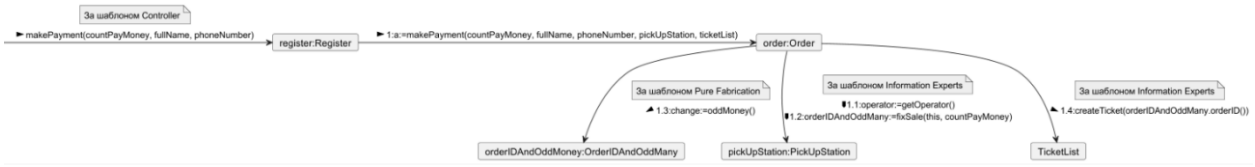

Рисунок 4.5 - Проектне рішення makePayment

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

У першому був застосований шаблон проектування Create через потребу у створені об’єкту для замовлення і взаємодії х 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 здачі 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.

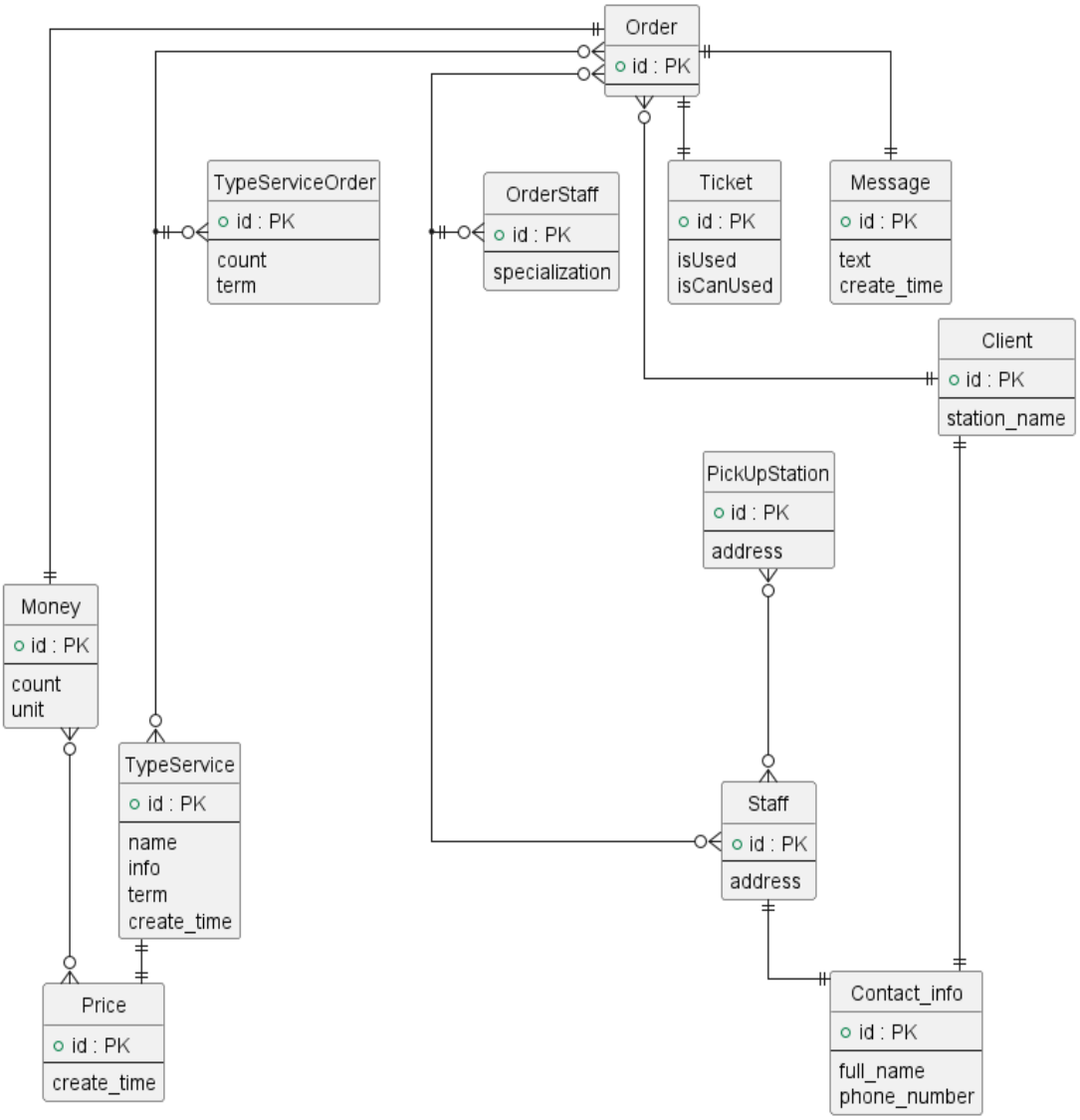

Рисунок 5.1: Концептуальні моделі даних для даного проекту

5.2 Реляційна модель даних

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

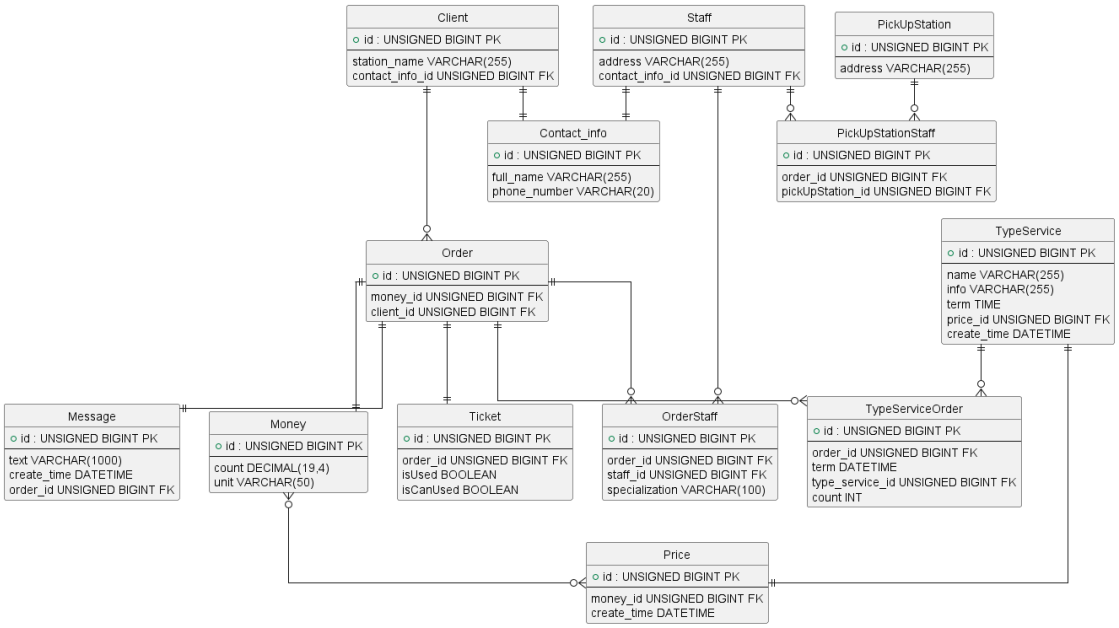

Рисунок 5.2. Реляційна модель даних для даного проекту

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

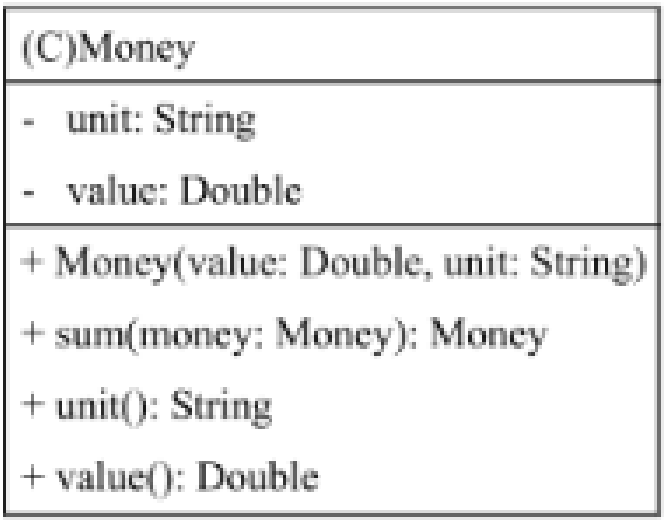

Рисунок 6.1 - Специфікація програмного класу Money

6.2 Специфікація програмного класу Operator

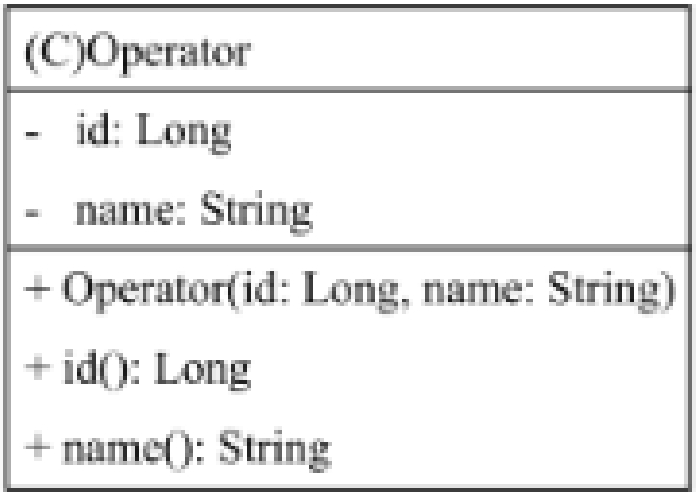

Рисунок 6.2 - Специфікація програмного класу Operator

6.3 Специфікація програмного класу Order

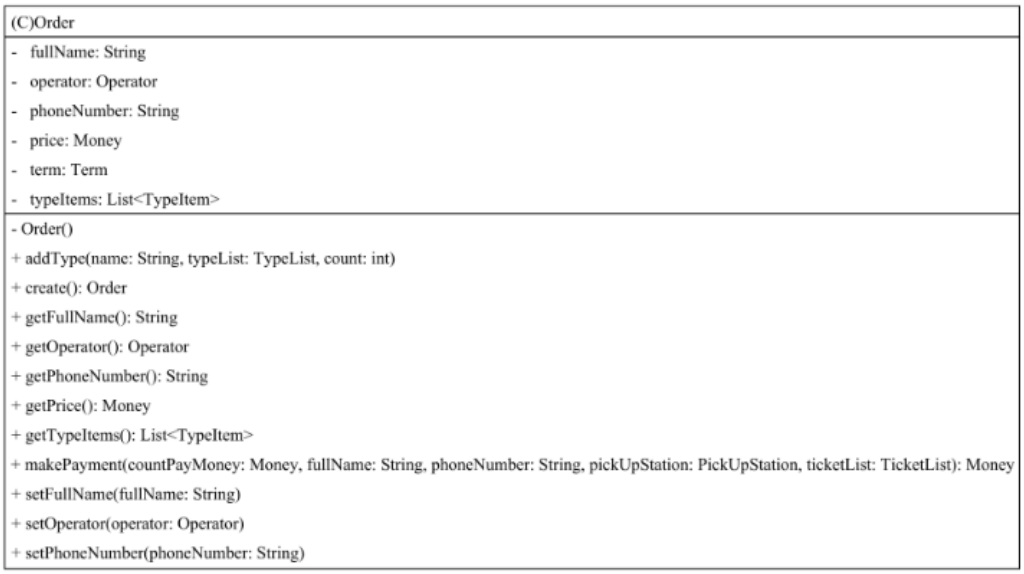

Рисунок 6.3 - Проектне рішення Order

6.4 Специфікація програмного класу OrderIDAndOddMoney

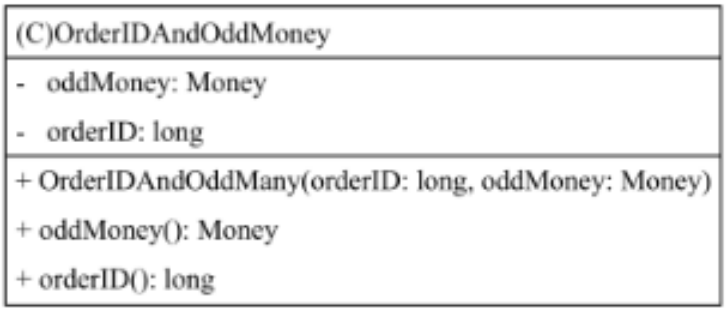

Рисунок 6.4 - Проектне рішення OrderIDAndOddMoney

6.5 Специфікація програмного класу PickUpStation

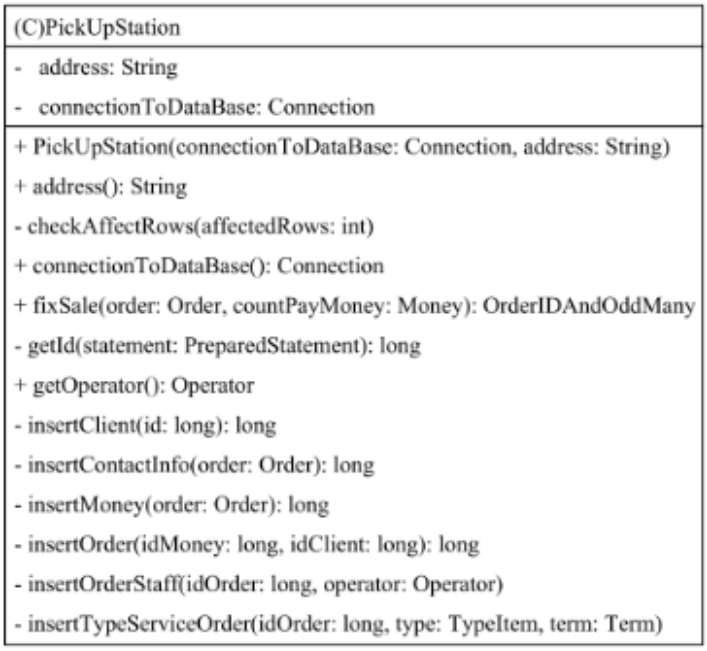

Рисунок 6.5 - Специфікація програмного класу TypeItem

6.6 Специфікація програмного класу Register

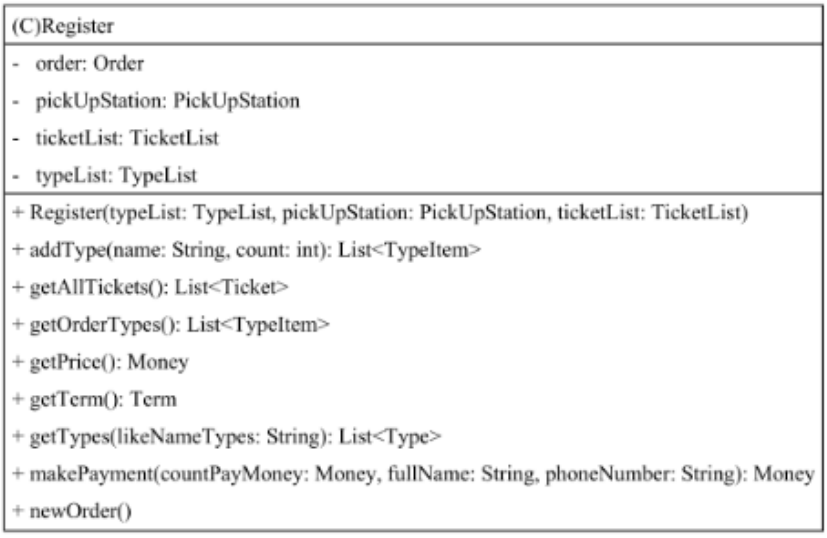

Рисунок 1.6 - Специфікація програмного класу Register

6.7 Специфікація програмного класу Term

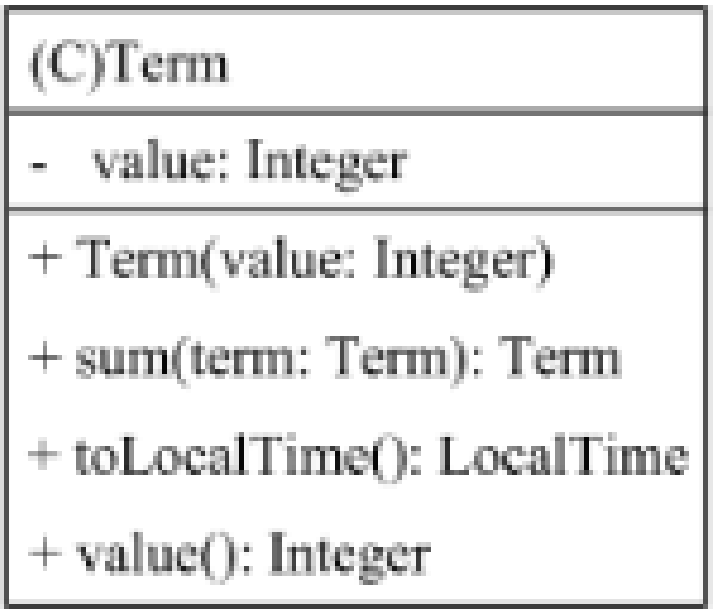

Рисунок 6.7 - Специфікація програмного класу Term

6.8 Специфікація програмного класу Ticket

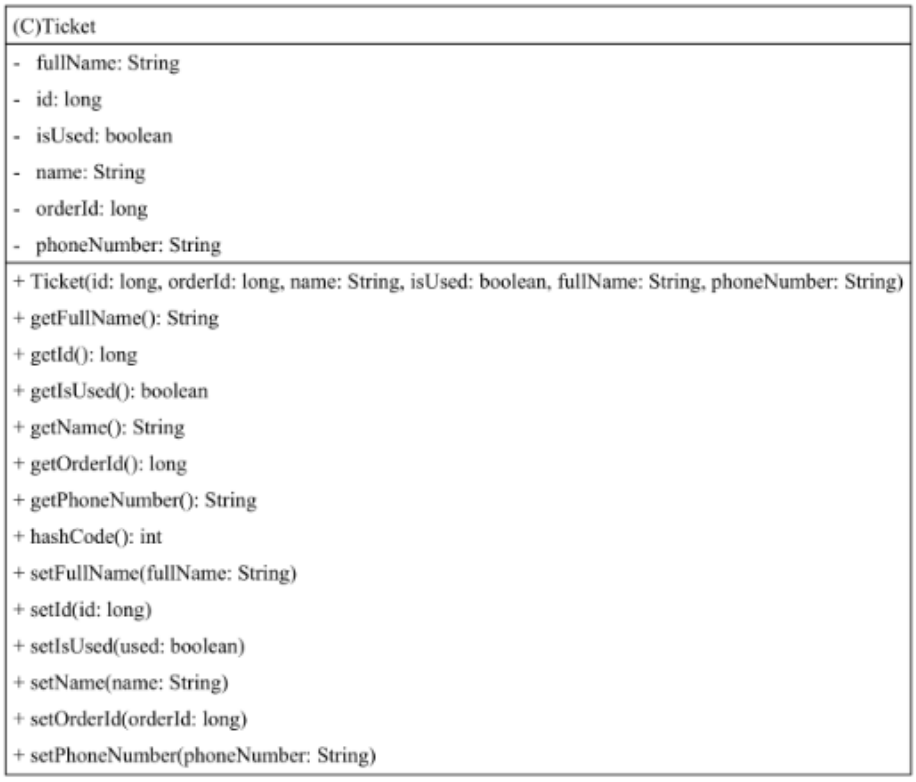

Рисунок 6.8 - Специфікація програмного класу Ticket

6.9 Специфікація програмного класу TicketList

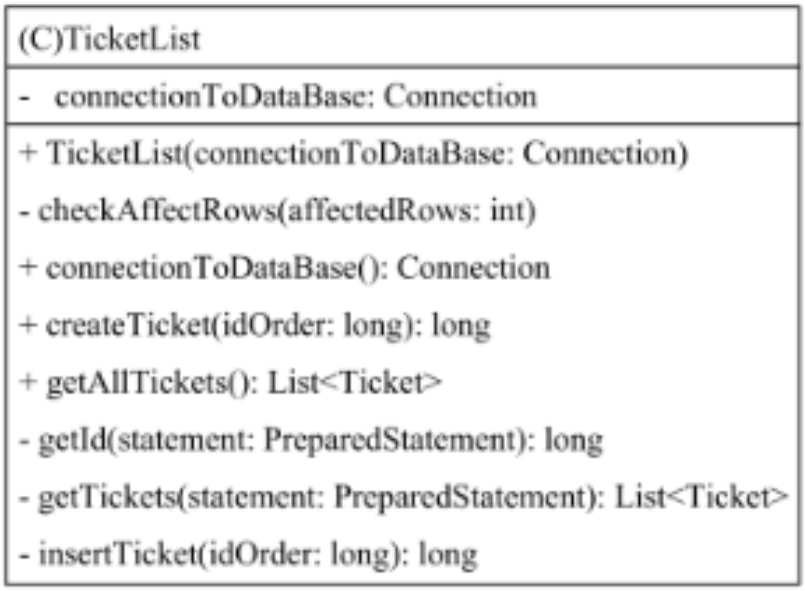

Рисунок 6.9 - Специфікація програмного класу TicketList

6.10 Специфікація програмного класу Type

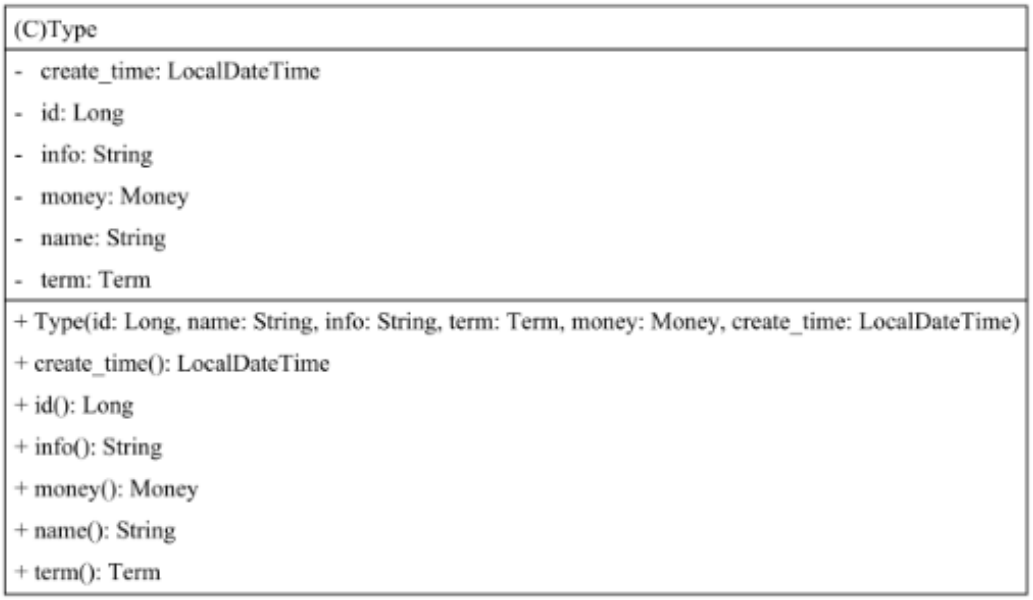

Рисунок 6.10 - Специфікація програмного класу Type

6.11 Специфікація програмного класу Type

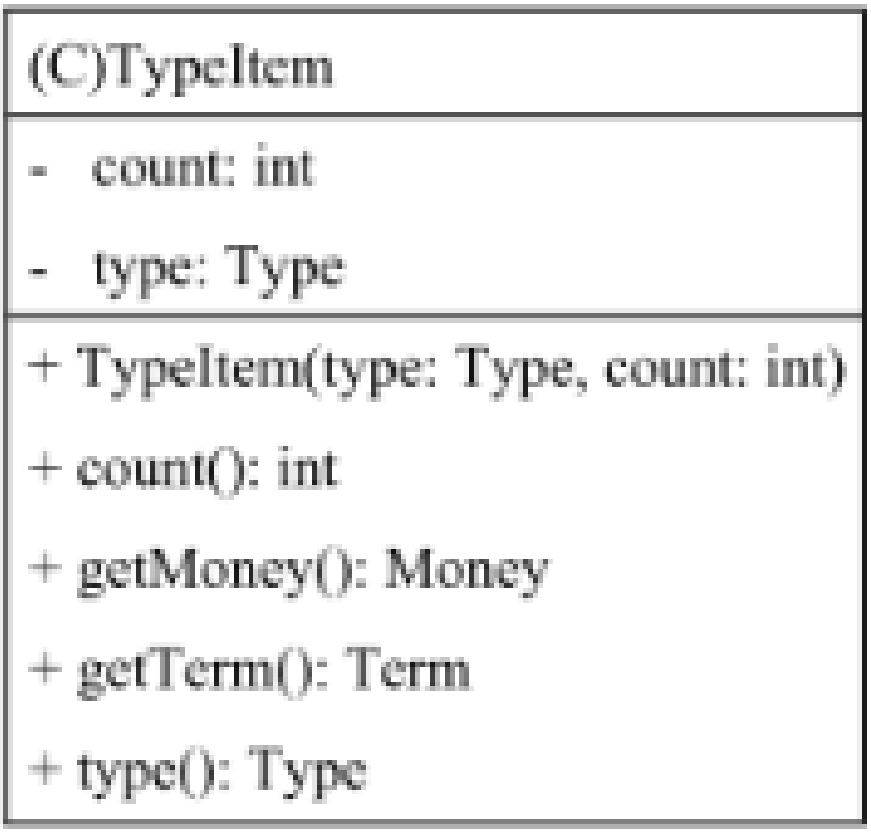

Рисунок 6.11 - Специфікація програмного класу Type

6.12 Специфікація програмного класу TypeList

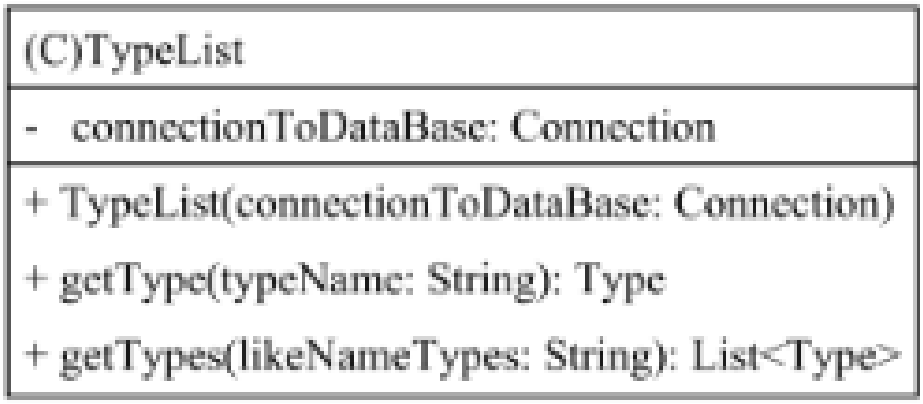

Рисунок 6.12 - Специфікація програмного класу TypeList

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

Для класу Register методи випливають 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 випливають 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 Type методи з атрибутами випливають 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 випливають 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 випливають 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 випливають 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 випливають 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.

Через аналіз відношень класів, самих змісту пунктів першого та другого прецеденту, проектних рішень, а також специфікацій було отримано наступна діаграма програмних класів на рисунку 6.13.

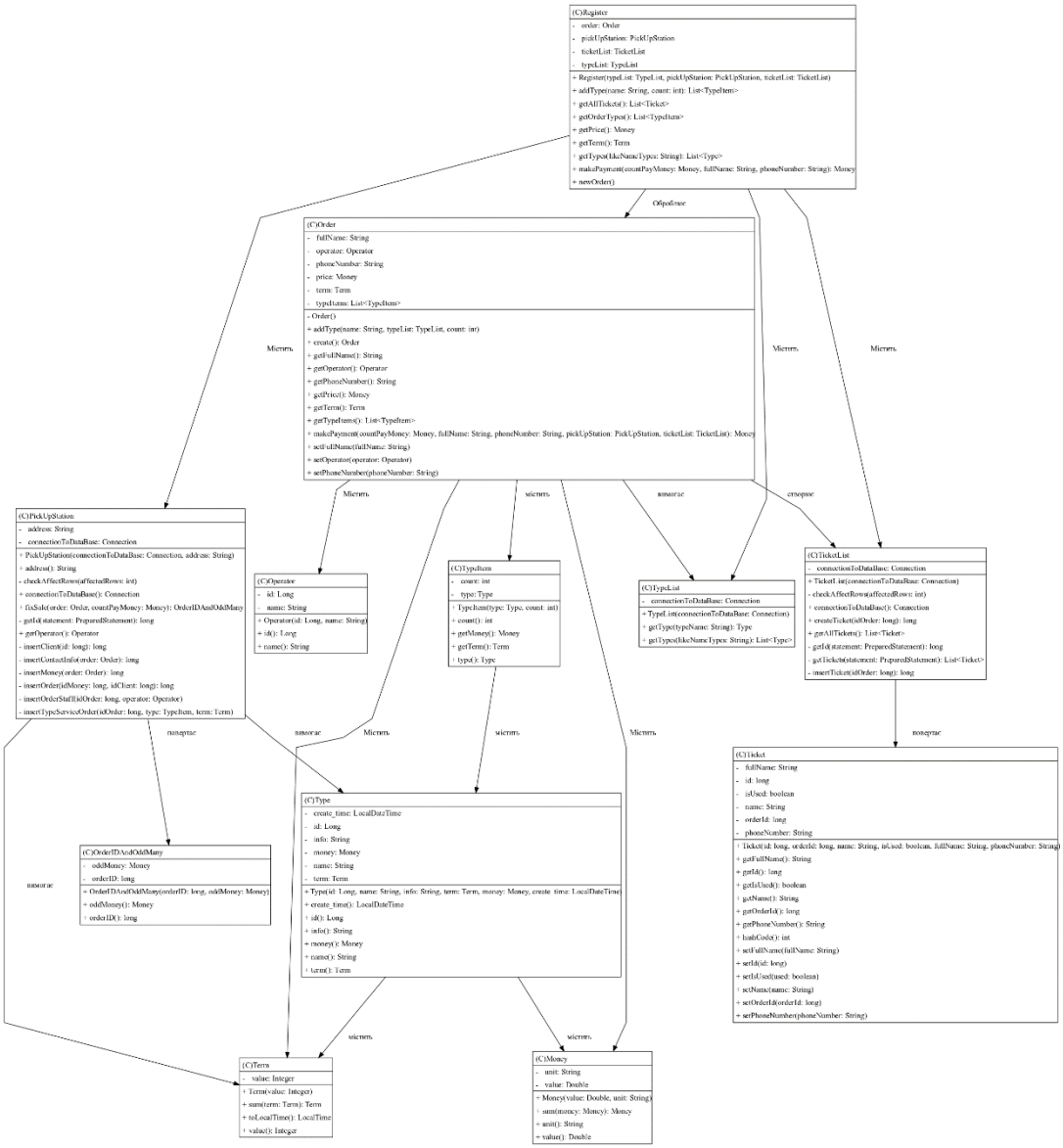

Рисунок 6.13 - Діаграма програмних класів

6.14 обґрунтування виявлених відношень між класами:

Для класу Register методи випливають 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 випливають 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 випливають 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 випливають 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 випливають 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 випливають 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 випливають 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

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                                    ss
on ss.staff_id = Staff.id
where ss.staff_id  is null;
```

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

Тест-кейси для методу getTypes()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC 1	Перевірка 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.dbDateTimeToLocalDateTime ("2024-05-27 11:48:27"))
TC 2	Виняток при відсутності типу	"незвичайний тип"	NotExistTypeException

Тест-кейси для методу addType()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC 1	Перевірка додавання нового типу	"Копіювання"	Type(2L, "Копіювання", "Кольорове копіювання", новий Term(150), новий Money(200.0000, "UAH"), Mysql.dbDateTimeToLocalDateTime ("2024-05-27 11:48:24"))

Тест-кейси для методу getAllTickets()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат

TC1	Перевірка отримання всіх квитків	Немає	Список квитків з одним елементом: Ticket(1, 1, "Друк", false, "Іван Іванов", "+380501234567")
TC2	Перевірка відсутності квитків	Порожня база даних	Порожній список квитків
TC3	Перевірка наявності кількох квитків	Декілька квитків у базі даних	Список квитків відповідно до записів у базі даних

Тест-кейси для методу createTicket()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC1	Створення нового квитка для існуючого замовлення	idOrder = 1 (існуюче замовлення)	Новий квиток створено, idTicket збігається з записом у базі даних
TC2	Створення нового квитка для неіснуючого замовлення	idOrder = 9999 (неіснуюче замовлення)	Виняток або відповідна помилка при створенні квитка
TC3	Перевірка створення кількох квитків	idOrder = 1 (існуюче замовлення), кілька викликів методу	Створення кількох квитків з унікальними idTicket для одного замовлення

Тест-кейси для методу getTerm()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC1	Перевірка обчислення терміну для доданого типу	"Друк", 57 хвилин	Термін 00:57:00
TC2	Перевірка обчислення терміну для іншого типу	"Копіювання", 30 хвилин	Термін 00:07:30
TC3	Перевірка обчислення терміну для декількох типів	"Друк", 30 хвилин і "Копіювання", 20 хвилин	Термін 00:50:00

Тест-кейси для методу getPrice()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
TC1	Перевірка обчислення ціни для доданого типу	"Друк", 57 одиниць	Ціна 5700.0000 UAH
TC2	Перевірка обчислення ціни для іншого типу	"Копіювання", 30 одиниць	Ціна 7500.0000 UAH
TC3	Перевірка обчислення ціни для декількох типів	"Друк", 30 одиниць і "Копіювання", 20 одиниць	Ціна 3000.0000 UAH

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

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")
ТС2	Перевірка відсутності оператора для неіснуючої станції	Станція "Неіснуюча адреса"	Виняток або null

Тест-кейси для методу fixSale()

Тест-кейс	Опис	Вхідні дані	Очікуваний результат
ТС1	Перевірка завершення продажу та розрахунку здачі	Замовлення з типом "Друк", кількість 3, оплата 500 UAH	Здача 200 - (ціна замовлення), запис у базі даних з правильним ім'ям та ідентифікатором
ТС2	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

ТС3	Перевірка завершення продажу з недостатнім платежем	Замовлення з типом "Друк", кількість 3, оплата 150	Виняток або відповідна помилка

ВИСНОВКИ

У даній курсової роботі були поглибленні та закріпленні 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 запроси 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 протестована 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

Register.java

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

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;

    public Register(TypeList typeList, PickUpStation pickUpStation, TicketList ticketList) {
        this.typeList = typeList;
        this.pickUpStation = pickUpStation;
        this.ticketList = ticketList;
    }

    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 likeNameTickets) throws
NotExistTypeException {
        return typeList.getTypes(likeNameTickets);
    }

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

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

```
    }
}
```

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() {
        price = typeItems.stream()
                .map(TypeItem::getMoney)
                .reduce(Money::sum)
                .orElseThrow();
        return price;
    }
    public Money makePayment(Money countPayMoney,
                             String fullName,
                             String phoneNumber,
                             PickUpStation pickUpStation,
                             TicketList ticketList) {
        this.fullName = fullName;
        this.phoneNumber = phoneNumber;
        operator = pickUpStation.getOperator();

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

```
}
```

OrderIDAndOddMoney.java

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

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

PickUpStation.java

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

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

public record PickUpStation(Connection connectionToDataBase, String address) {


    /*
     * Через запрос у базу даних отримуються вільний оператор
     * та повератється 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 здачу. (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 принтеру');\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.");
            }
        }
    }
```

Term.java

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

import java.time.LocalTime;
import java.time.format.DateTimeFormatter;

public record Term(Integer value) {

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

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

    @Override
    public String toString() {
        LocalTime 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 isUsed;
private String fullName;
private String 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 getIsUsed() {
    return isUsed;
}

public void setIsUsed(boolean used) {
    isUsed = used;
}

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;
}

public Ticket(long id,
              long orderId,
              String name,
              boolean isUsed,
              String fullName,
```

```
            String phoneNumber) {
    this.id = id;
    this.orderId = orderId;
    this.name = name;
    this.isUsed = isUsed;
    this.fullName = fullName;
    this.phoneNumber = phoneNumber;
}



@Override
public boolean equals(Object obj) {
    if (obj == this) return true;
    if (obj == null || obj.getClass() != this.getClass()) return false;
    var that = (Ticket) obj;
    return this.id == that.id &&
            this.orderId == that.orderId &&
            Objects.equals(this.name, that.name) &&
            this.isUsed == that.isUsed &&
            Objects.equals(this.fullName, that.fullName) &&
            Objects.equals(this.phoneNumber, that.phoneNumber);
}

@Override
public int hashCode() {
    return Objects.hash(id, orderId, name, isUsed, fullName, phoneNumber);
}

@Override
```

```
    public String toString() {
        return "Ticket[" +
                "id=" + id + ", " +
                "orderId=" + orderId + ", " +
                "name=" + name + ", " +
                "isUsed=" + isUsed + ", " +
                "fullName=" + fullName + ", " +
                "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 List<Ticket> getAllTickets() {
        try {
            String sql = """
                    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;
                """;

            PreparedStatement statement =
connectionToDataBase.prepareStatement(sql);

            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("isUsed"),
                    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

Controller/OrderManagementController.java

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

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.vitapasser.photocopypoint.Controller.CreatingOrder.Controller;
import org.vitapasser.photocopypoint.MainApplication;
import org.vitapasser.photocopypoint.Model.Register;

import java.io.IOException;
import java.util.Objects;

public class OrderManagementController {

    private Register register;

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

        FXMLLoader FXMLLoader = new FXMLLoader(Objects.requireNonNull(
                MainApplication.class.getResource("creating-order.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.show();
    }
    public void initialize() {}
    public void putData(Register register) {
        this.register = register;
    }
}
```

Controller/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.CreatingOrder.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());
        OrderManagementController controller = FXMLLoader.getController();
        controller.putData(register);
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        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());
        Controller controller = FXMLLoader.getController();
        controller.putData(register);
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        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<>("isUsed"));

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

                changeTable();
            });
        }
```

```
        public void putData(Register register, String nameClient, String
phoneNumberClient) {
            this.register = register;
            this.nameClient = nameClient;
            this.phoneNumberClient = phoneNumberClient;
        }
    }
```

Controller/CreatingOrder/TypeView.java

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

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

import java.time.LocalTime;

public class TypeView {
    long id;
    String name;
    String info;
    LocalTime 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 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;
    }
}
```

Controller/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.OrderManagementController;
import org.vitapasser.photocopypoint.Controller.PaymentController;
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.LocalTime;
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.name);
    }

    @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 onCancelTypeServiceButtonClick(ActionEvent event) {
    }

    @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.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());
        OrderManagementController controller = FXMLLoader.getController();
        controller.putData(register);
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        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
PropertyValueFactory<>("name"));
            nameTypeColumn1.setCellValueFactory(new
PropertyValueFactory<>("name"));

            infoTypeColumn.setCellValueFactory(new
PropertyValueFactory<>("info"));
```

```
                infoTypeColumn1.setCellValueFactory(new
PropertyValueFactory<>("info"));

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

                countMoneyTypeColumn.setCellValueFactory(new
PropertyValueFactory<>("countMoney"));
                countMoneyTypeColumn1.setCellValueFactory(new
PropertyValueFactory<>("countMoney"));

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

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

                changeTable(typeViewsListOfSelectableServices);
                initLoadTackedTable();

            });

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

```
}
```

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" />

```
                        <Button contentDisplay="BOTTOM" disable="true"
mnemonicParsing="false" onAction="#onCancelTypeServiceButtonClick"
prefWidth="98.0" text="Відмінити" />
                      </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>
```

```
<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 обираємих послуг"
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>
```

Order-management.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.layout.BorderPane?>

<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.OrderManagementController">
    <right>
        <BorderPane prefWidth="339.0" BorderPane.alignment="CENTER">
            <left>
                <Separator orientation="VERTICAL" BorderPane.alignment="CENTER" />
            </left>
            <BorderPane.margin>
                <Insets />
            </BorderPane.margin>
        <center>
                <Button alignment="CENTER" mnemonicParsing="false" onAction="#onStartCreateOrderButtonClick" prefHeight="25.0" prefWidth="200.0" text="Створити нове замовлення" BorderPane.alignment="CENTER" />
        </center></BorderPane>
    </right>
```

```
</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">
                  <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" />
                        <TableColumn fx:id="isReadyColumn" prefWidth="75.0" text="Готово" />
                      </columns>
                      <columnResizePolicy>
                        <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                      </columnResizePolicy>
                    </TableView>
                  </content>
                </ScrollPane>
              </center>
              <top>
                <Label text="Послуги у замовлені" BorderPane.alignment="CENTER" />
              </top>
            </BorderPane>
          </center>
```

```
</BorderPane>
```

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.OrderManagementController;
import org.vitapasser.photocopypoint.Model.PickUpStation;
import org.vitapasser.photocopypoint.Model.Register;
import org.vitapasser.photocopypoint.Model.TicketList;
import org.vitapasser.photocopypoint.Model.TypeList;
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);

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

                FXMLLoader FXMLLoader = new
FXMLLoader(Objects.requireNonNull(
                        MainApplication.class.getResource("order-management.fxml")));
                Scene scene = new Scene(FXMLLoader.load());
                OrderManagementController controller = FXMLLoader.getController();
                controller.putData(register);
                stage.setTitle("Hello!");
                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();
    }
}
```

PickUpStationTest.java

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

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.vitapasser.photocopypoint.Util.Mysql;

import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Objects;

import static org.junit.jupiter.api.Assertions.*;
```

```
class PickUpStationTest {

    Connection connection = Mysql.getConnection();

    PickUpStationTest() throws SQLException, IOException {
    }

    @Test
    void getOperator() {
        PickUpStation pickUpStation = new PickUpStation(connection,
                "Київ, вул. Хрещатик, 1");
        Operator operatorGet = pickUpStation.getOperator();

        Operator operatorReference = new Operator(2L, "Петро Петренко");

        Assertions.assertEquals(operatorReference, operatorGet);
    }

    @Test
    void fixSale() {
        PickUpStation pickUpStation = new PickUpStation(connection,
                "Київ, вул. Хрещатик, 1");
        Order order = Order.create();

        TypeList typeList = new TypeList(connection);
        order.addType("Друк", typeList, 3);
        order.getTerm();
        order.getPrice();
        String fullName = "ТЕСТДорно ДованоТЕСТ";
        order.setFullName(fullName);
```

```
            order.setPhoneNumber("+3809998748904");
            order.setOperator(new Operator(1L, "Петро Петренко"));

            Money pay = new Money(500.00, "UAH");

            OrderIDAndOddMany orderIDOddMoney = pickUpStation.fixSale(order,
pay);
            assert orderIDOddMoney != null;
            Money odd_money = orderIDOddMoney.oddMoney();
            Money odd_money_check = new Money(pay.value() - order.getPrice().value(),
pay.unit());

            Boolean oddMoneyTest = Objects.equals(odd_money, odd_money_check);

            String sqlCheck = "SELECT `PhotocopyPoint`.`Contact_info`.`full_name`,
`PhotocopyPoint`.`Order`.id FROM `PhotocopyPoint`.`Contact_info`\n" +
                    "INNER JOIN `PhotocopyPoint`.Client on
`PhotocopyPoint`.Client.contact_info_id = `PhotocopyPoint`.`Contact_info`.`id`\n" +
                    "INNER JOIN `PhotocopyPoint`.`Order` on
`PhotocopyPoint`.`Order`.`client_id` = `PhotocopyPoint`.Client.id\n" +
                    "WHERE `PhotocopyPoint`.`Contact_info`.`full_name` =
'"+fullName+"'\n" +
                    "GROUP BY `PhotocopyPoint`.`Order`.id ORDER BY
`PhotocopyPoint`.`Order`.id DESC LIMIT 1;";

            String fullNameCreated = null;
            long orderId = -1;
            try {
                Statement statement = connection.createStatement();
```

```
            ResultSet sqlResult = statement.executeQuery(sqlCheck);
            sqlResult.next();

            fullNameCreated = sqlResult.getString("full_name");
            orderId = sqlResult.getLong("id");


        } catch (Exception e)
        {
            System.out.println("Error on test 'FixSale': " +e.getMessage());
            e.printStackTrace();
        }

        Assertions.assertEquals(fullNameCreated, fullName);
        Assertions.assertEquals(orderId, orderIDOddMoney.orderID());
        Assertions.assertEquals(odd_money, odd_money_check);
    }
}
```

RegisterTest.java

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

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.vitapasser.photocopypoint.Util.Mysql;

import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Objects;
```

```
import static org.junit.jupiter.api.Assertions.*;

class RegisterTest {

    Connection connection = Mysql.getConnection();

    RegisterTest() throws SQLException, IOException {
    }

    @Test
    void getTerm() {
        TypeList typeList = new TypeList(connection);
        PickUpStation pickUpStation = new PickUpStation(connection,
                "Київ, вул. Хрещатик, 1");
        TicketList ticketList = new TicketList(connection);

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

        register.newOrder();
        register.addType("Друк", 57);
        Term term = register.getTerm();

        Term testTerm = new Term(57 * 60);

        Assertions.assertEquals(testTerm, term);
    }

    @Test
    void getPrice() {
```

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

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

    register.newOrder();
    register.addType("Друк", 57);
    Money price = register.getPrice();

    Money testPrice = new Money(57 * 100.0000, "UAH");

    Assertions.assertEquals(testPrice, price);
}

@Test
void makePayment() {
    TypeList typeList = new TypeList(connection);
    PickUpStation pickUpStation = new PickUpStation(connection,
            "Київ, вул. Хрещатик, 1");
    TicketList ticketList = new TicketList(connection);

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

    register.newOrder();
    register.addType("Друк", 57);
    register.getTerm();
    register.getPrice();
```

```
        Money oddMoney = register.makePayment(new Money(60 * 100.0000,
"UAH"),
                "Сергій Сергійович Сержі",
                "+3809844678983");

        Money testOddMoney = new Money(3 * 100.0000, "UAH");

        Assertions.assertEquals(testOddMoney, oddMoney);
    }
}
```

TicketList.java

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

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.vitapasser.photocopypoint.Util.Mysql;

import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

class TicketListTest {

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

```
    TicketListTest() throws SQLException, IOException {
    }

    @Test
    void getAllTickets() {
        TypeList typeList = new TypeList(connection);
        PickUpStation pickUpStation = new PickUpStation(connection,"Київ, вул. Хрещатик, 1");
        TicketList ticketList = new TicketList(connection);
        Register register = new Register(typeList, pickUpStation, ticketList);

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

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

        ticketReference.add(new Ticket(1, 1, "Друк", false, "Іван Іванов", "+380501234567"));

        Assertions.assertEquals(ticketReference.getFirst(), tickets.getFirst());
    }

    @Test
    void createTicket() {
        TicketList ticketList = new TicketList(connection);

        long idOrder = 1;
        long idTicket = ticketList.createTicket(idOrder);

        String sqlCheck = "SELECT `PhotocopyPoint`.`Order`.id as OrderId, `PhotocopyPoint`.`Ticket`.id as TicketId\n" +
```

```
                    "FROM `PhotocopyPoint`.`Ticket`\n" +
                    "INNER JOIN `PhotocopyPoint`.`Order` " +
                    "on `PhotocopyPoint`.`Order`.`id` =
`PhotocopyPoint`.Ticket.order_id\n" +
                    "WHERE `PhotocopyPoint`.`Order`.`id` = "+idOrder+"\n" +
                    "GROUP BY OrderId, TicketId " +
                    "ORDER BY OrderId, TicketId DESC LIMIT 1;";

            long testOrderId = -1, testTicketId = -1;
            try {
               Statement statement = connection.createStatement();

               ResultSet sqlResult = statement.executeQuery(sqlCheck);
               sqlResult.next();

               testOrderId = sqlResult.getLong("OrderId");
               testTicketId = sqlResult.getLong("TicketId");

            } catch (Exception e)
            {
               System.out.println("Error on test 'FixSale': " +e.getMessage());
               e.printStackTrace();
            }

            Assertions.assertEquals(idOrder, testOrderId);
            Assertions.assertEquals(idTicket, testTicketId);
         }
      }
```

TypeListTest.java

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

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.vitapasser.photocopypoint.Exception.NotExistTypeException;
import org.vitapasser.photocopypoint.Util.Mysql;

import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

class TypeListTest {

    Connection connection = Mysql.getConnection();

    TypeListTest() throws SQLException, IOException {
    }

    @Test
    void getTypes(){
        TypeList typeList = new TypeList(connection);
        List<Type> typesGet;
        try {
            typesGet = typeList.getTypes("д");
        } catch (NotExistTypeException e) {
            throw new RuntimeException(e);
        }
```

```
        List<Type> typesReference = new ArrayList<>();

        typesReference.add(new Type(5L, "Друк", "Чорно-білий друк",
                new Term(60), new Money(100.0000, "UAH"),
Mysql.dbDateTimeToLocalDateTime("2024-05-27 11:48:27")));

        Assertions.assertEquals(typesReference, typesGet);

    }

    @Test
    void addType(){
        TypeList typeList = new TypeList(connection);
        Type typeGot = typeList.getType("Копіювання");

        Type typeReference = new Type(2L, "Копіювання", "Кольорове
копіювання",
                new Term(150), new Money(200.0000, "UAH"),
Mysql.dbDateTimeToLocalDateTime("2024-05-27 11:48:24"));

        Assertions.assertEquals(typeGot, typeReference);

    }
}
```

Modul-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;
}
```

ДОДАТОК В СТРУКТУРА ФАЙЛІВ КОДУ ПРОГРАМИ

На рисунку .1 та .2 зображена структура файлів код і папок проекту.

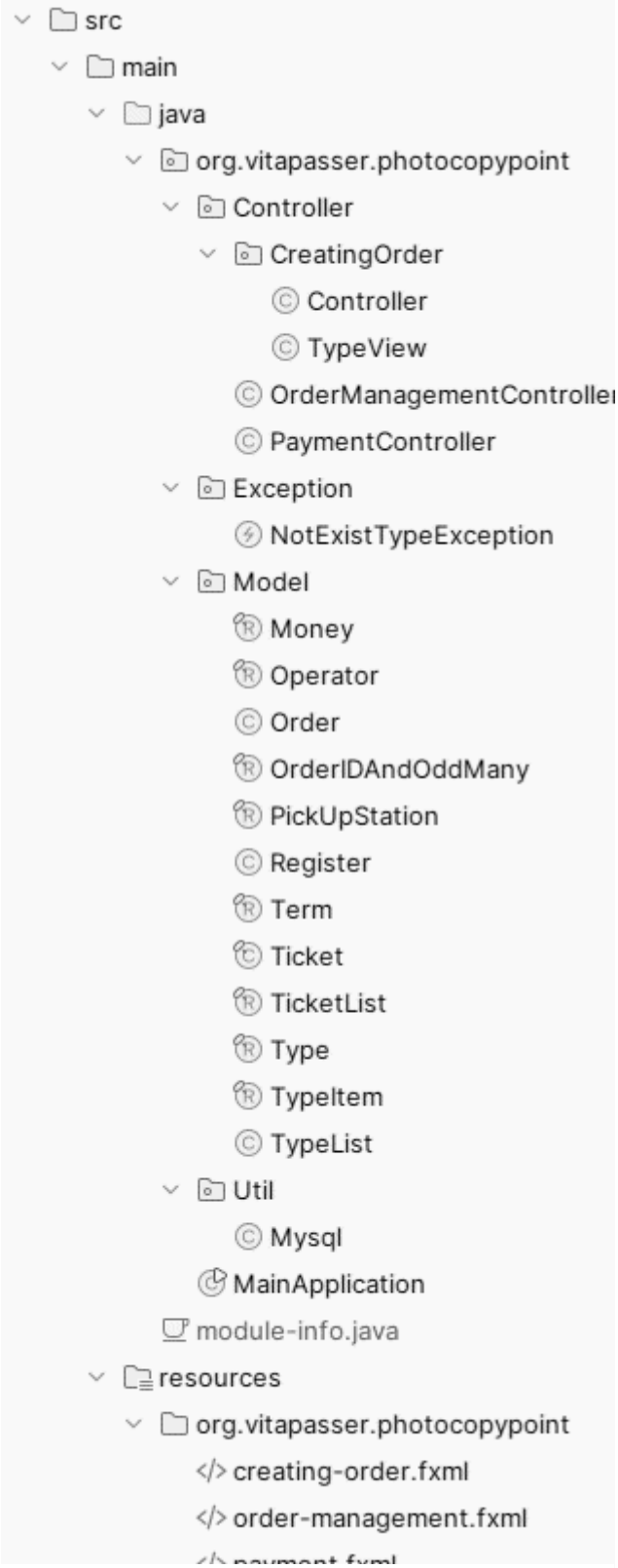

Рисунок 1.Ошибка! Текст указанного стиля в документе отсутствует.**.1 - Структура файлів і папок проекту.**

Рисунок 1.2 - Продовження структури файлів та 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>
    </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.MainApplicatio
n</mainClass>
                  <launcher>app</launcher>
                  <jlinkZipName>app</jlinkZipName>
                  <jlinkImageName>app</jlinkImageName>
                  <noManPages>true</noManPages>
                  <stripDebug>true</stripDebug>
                  <noHeaderFiles>true</noHeaderFiles>
                </configuration>
              </execution>
            </executions>
          </plugin>
        </plugins>
      </build>
```

```
</project>
```