МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп’ютерних наук та управління проєктами

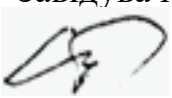
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

 проф. Приходько С.Б.

«__» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

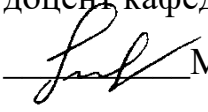
на тему: Розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»

Виконав: студент групи 4151

 Білошицький М. П.

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

 Макарова Л. М

м. Миколаїв – 2026 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_07_» ___04___ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Білошицькому Максиму Павловичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech» _____
_____

Керівник роботи: доцент кафедри ПЗАС, к.т.н., доцент Макарова Л. М. _____
Затверджені наказом ректора №275-уч від 07.04.2026 р.
2. Термін подання роботи: 01.06.2026 р. _____
3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів _____
Титульний слайд, Актуальність та мета роботи, Огляд існуючих програмних рішень, Функціональні вимоги та рольова модель, Діаграма варіантів використання, Діаграма класів, Архітектура програмного забезпечення, Діаграма послідовності, Технологічний стек та обґрунтування вибору, Фізична модель даних, Інтерфейс користувача застосунку РемTech, Тестування програмного забезпечення, Результати розробки, Висновки, Заключний слайд. _____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 07.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	30.03.2026	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	06.04.2026	ПП*
3.	Аналіз вимог до ПЗ	13.04.2026	ПП*
4.	Розробка архітектури ПЗ	20.04.2026	
5.	Розробка проекту ПЗ	04.05.2026	
6.	Реалізація та тестування ПЗ	11.05.2026	
7.	Підготовка розділу Результати розробки ПЗ	15.05.2026	
8.	Підготовка розділу з охорони праці	18.05.2026	ПП*
9.	Підготовка розділу ВИСНОВКИ	22.05.2026	
10.	Оформлення списку використаних джерел та додатків	25.05.2026	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	01.06.2026	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	05.06.2026	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	08.06.2026	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	15.06.2026	

* - за результатами переддипломної практики (ПП), яка була з 02.02.2026 до 22.03.2026 р.

Студент _____ Білошицький М. П.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме задачі розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

У процесі роботи було проведено аналіз практичної задачі інженерії програмного забезпечення, здійснено огляд і порівняння існуючих програмних рішень у сфері автоматизації діяльності сервісних центрів, сформульовано постановку задачі, визначено функціональні та нефункціональні вимоги для розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech». Представлено проєкт та результати розробки програмного забезпечення. Також виконано розділ з охорони праці.

Робота викладена на 119 сторінках, містить 21 рисунок, 10 таблиць, список використаних джерел із 29 найменувань та 5 додатків.

ABSTRACT

The qualification work is devoted to solving a practical task of software engineering characterized by the complexity and uncertainty of conditions, namely the task of developing software for automating the accounting of the work performance of a computer repair service center "РемTech".

An analysis of the practical software engineering problem was carried out, as well as a review and comparison of existing software solutions in the field of service center automation. The problem statement was formulated, and both functional and non-functional requirements for the development of software for automating the accounting of the work performance of a computer repair service center "РемTech" were defined. The project and the results of the software development are presented. In addition, special section on labor protection were developed.

Work is presented on 119 pages and contains 21 figures, 10 tables, a list of references including 29 sources, and 5 appendices.

ЗМІСТ

ВСТУП

Сучасний етап розвитку сфери послуг характеризується стрімкою цифровізацією та підвищенням вимог до швидкості та якості обслуговування клієнтів. Сервісні центри з ремонту комп'ютерної та оргтехніки щодня опрацьовують велику кількість замовлень, кожне з яких потребує чіткого контролю. Відсутність автоматизованого програмного забезпечення для обліку виконання робіт призводить до втрати даних, помилок у розрахунках та зниження лояльності клієнтів.

Останніми роками ІТ-галузь демонструє стабільне зростання та залишається ключовим драйвером цифрової трансформації бізнесу, що підтверджується аналітичними звітами Gartner [1].

Водночас світовий ринок персональних комп’ютерів демонструє відновлення після спаду. Зокрема, поставки ПК у 2025 році зросли приблизно на 8–9% у річному вимірі [2], що свідчить про збільшення кількості техніки в експлуатації та, відповідно, підвищення потреби в її обслуговуванні та ремонті.

Станом на 2025 рік ІТ-галузь України демонструє поступове відновлення та зростання після кризових періодів: зростає експорт комп’ютерних послуг, що свідчить про збереження високого попиту на технологічні рішення [3]. Для малого та середнього бізнесу, що працює у сфері ремонту комп’ютерної техніки, впровадження сучасних програмних інструментів автоматизації є особливо актуальним в умовах сучасного українського ринку.

Створення та впровадження програмного забезпечення дозволяє централізувати збереження інформації, автоматизувати рутинні операції та формування звітності, а також забезпечити розмежування прав доступу між персоналом, що є критично важливим для стабільного функціонування бізнесу та конкурентоспроможності на ринку України.

Існує значна кількість програмних продуктів для автоматизації обліку діяльності сервісних центрів, проте більшість із них є або універсальними рішеннями, або орієнтованими на конкретні підприємства, що не враховують специфіку бізнес-процесів окремого сервісного центру. Такі рішення часто мають надлишковий або, навпаки, недостатній функціонал та потребують складної адаптації. Тому такі рішення можуть не забезпечувати належного рівня автоматизації, або є економічно недоцільними через високу вартість та надлишкову функціональність. Розробка власного програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech» дозволить врахувати специфіку бізнес-процесів підприємства.

Таким чином, розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech» є актуальною.

Кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме задачі розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech».

Комплексність та невизначеність умов розв'язання обраної задачі інженерії програмного забезпечення полягає у неповноті та неоднорідності вхідних даних, та впливу людського фактору на процеси обліку й управління, та зумовлена необхідністю інтеграції різних функціональних процесів сервісного центру, а саме: приймання та видачі замовлень, фіксування витрат та наявності запчастин, облік робочого часу майстрів, формування вартості ремонту на основі наданих послуг, формування звітності в межах єдиної програми обліку із забезпеченням цілісності та узгодженості даних, одночасної роботи із програмою та підтримки різних рівнів доступу для надання ефективної взаємодії між співробітниками, з

різними зонами відповідальності.

Розв’язання цієї задачі в межах кваліфікаційної роботи передбачає реалізацію програмного забезпечення, здатного пришвидшити обслуговування клієнтів, скоротити кількість помилок, підвищити зручність ведення обліку даних, покращити взаємодію працівників сервісного центру з ремонту комп’ютерної техніки «РемTech».

Метою кваліфікаційної роботи є розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Для досягнення поставленої мети кваліфікаційної роботи необхідно вирішити наступні завдання:

- провести аналіз практичної задачі інженерії програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»;
- провести огляд та аналіз існуючих програмних продуктів програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки;
- сформулювати постановку задачі та технічні вимоги на розробку програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»;
- провести аналіз вимог до програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»;
- розробити архітектуру програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»;
- розробити проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту техніки «РемTech».

- виконати тестування та випробування програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».
- розробити програмну документацію для програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту техніки «РемTech».

Кваліфікаційна робота передбачає повний процес розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Об’єктом кваліфікаційної роботи є процес розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАСТОСУНКУ «РЕМТЕСН»

1.1 Аналіз практичної задачі інженерії програмного забезпечення

В даній роботі розв’язується задача розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемТech».

Сервісний центр надає послуги з ремонту та обслуговування ноутбуків, нетбуків, персональних комп’ютерів, комп’ютерних комплектуючих та іншої комп’ютерної техніки. У процесі автоматизації діяльності сервісного центру розглядаються ключові процеси ремонту та обслуговування електронних пристроїв, а також взаємодія з клієнтами. До основних процесів належать:

- Управління клієнтською базою: ведення історії звернень, контактних даних.
- Прийом та діагностика: реєстрація пристрою, опис заявленої несправності та візуального стану.
- Управління замовленнями та ремонтами: призначення відповідального майстра та відстеження статусів виконання.
- Складський облік: контроль наявності запчастин та їх списання під конкретні ремонти.
- Фінансовий облік: розрахунок вартості послуг на основі наданих послуг та використаних комплектуючих.

Використання традиційних методів обліку, наприклад паперових журналів, неструктурованих таблиць Excel, створює низку критичних проблем. Неузгодженість даних про складські залишки призводить до нестачі необхідних

запчастин у критичний момент, вплив людського фактору під час ручного формування квитанцій зумовлює підвищення ризику виникнення помилок, зокрема фінансового, облікового та інформаційного характеру, а відсутність єдиної системи спричиняє проблему «зон відповідальності» – нечіткий розподіл обов’язків призводить до втрати актуальності даних та зниження якості сервісу.

Розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» покликана вирішити ці проблеми, та надати інструмент автоматизації повного циклу обслуговування клієнтів – від первинної реєстрації несправного пристрою та клієнта до формування звітності.

Клієнтська частина може бути представлена кількома однотипними екземплярами, які одночасно підключаються до серверної частини, де використовується клієнт-серверна архітектура із застосуванням REST та протоколу HTTP для обміну даними, що є поширеним підходом у CRM- та ERP-системах [4]. Вибір такого підходу зумовлений наступними факторами вимог практичної задачі інженерії програмного забезпечення:

- Централізація даних: всі дані зберігаються на виділеному сервері. Це виключає ризик розсинхронізації інформації, коли співробітники бачать різні статуси одного замовлення.
- Безпека та розмежування доступу: архітектура системи передбачає обов'язкову автентифікацію та чітку рольову модель авторизації (адміністратор, майстер, директор). Клієнтський застосунок обмежує доступ до певних модулів, функцій та конфіденційної інформації залежно від посади користувача, що мінімізує ризик несанкціонованого втручання в критичні бізнес-процеси та базу даних сервісного центру.
- Масштабованість: архітектура дозволяє легко підключати нові робочі місця без необхідності дублювання бази даних на кожному комп'ютері.

1.2 Огляд та аналіз існуючих програмних рішень для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки

Існує значна кількість програмних продуктів для автоматизації обліку діяльності сервісних центрів, проте більшість із них є або універсальними рішеннями, або орієнтованими на конкретні підприємства, що не враховують специфіку бізнес-процесів окремого сервісного центру. Такі рішення часто мають надлишковий або, навпаки, недостатній функціонал та потребують складної адаптації. Тому можуть не забезпечувати належного рівня автоматизації, або є економічно недоцільними через високу вартість та надлишкову функціональність.

«RO App» – це онлайн-сервіс (SaaS) для комплексного управління бізнесами в таких сферах, як автосервіси, продажі, послуги та ремонт електроніки. Застосунок надає інструменти для ведення складського обліку, управління заявками та завданнями майстрів, а також контролю платежів і зарплат. Завдяки цим можливостям, «RO App» допомагає підвищити ефективність управління всіма аспектами бізнесу [5].

Однак сервіс є платним, висока вартість регулярної підписки, яка зростає зі збільшенням кількості співробітників та локацій.

Система акаунтів, де вся інформація прив'язана до Google-акаунта, а усі комерційні дані та персональні дані клієнтів зберігаються на сторонніх серверах. Це створює ризик витоку даних або несанкціонованого доступу, якщо компанія-розробник не забезпечує належний рівень захисту. Використання акаунта Google для доступу до системи означає, що у випадку проблем із Google-акаунтом, бізнес може втратити доступ до своїх даних.

На рисунку 1.1 представлено зовнішній вигляд застосунку «RO App».

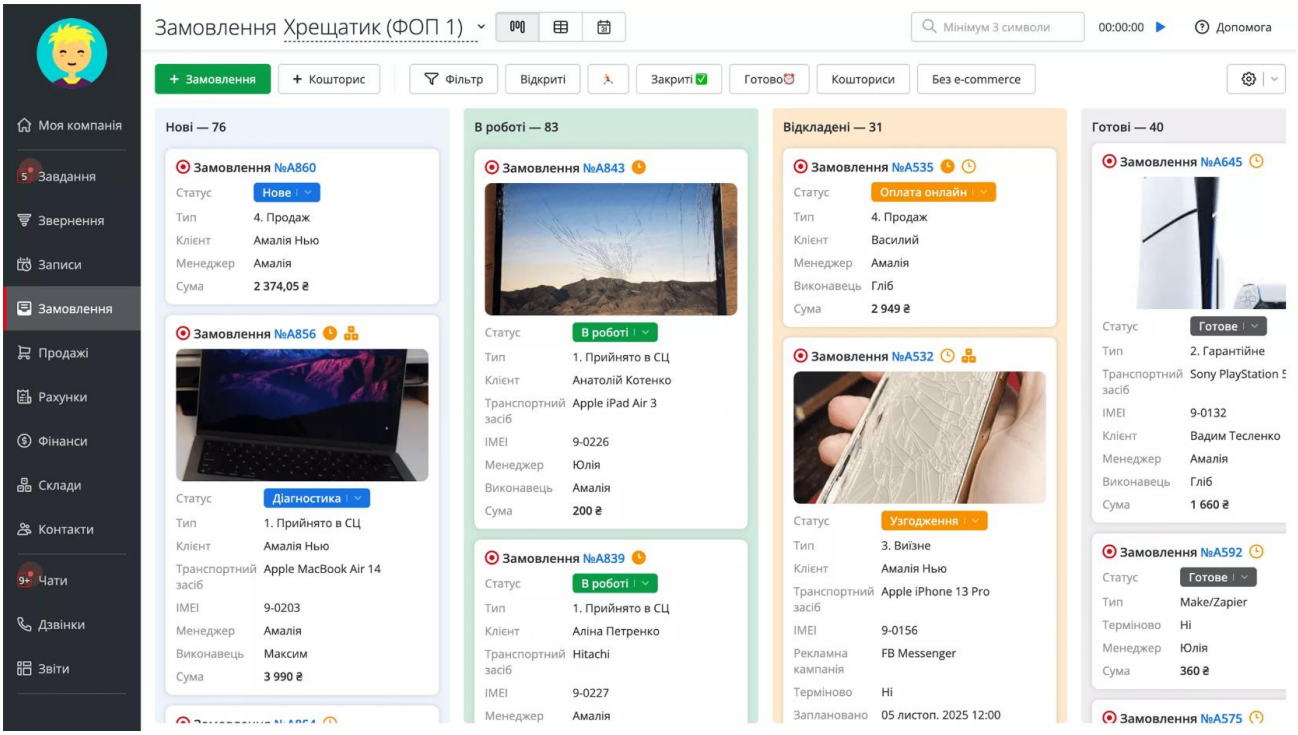


Рисунок 1.1 – Вкладка «Замовлення» у застосунку «RO App».

Також застосунок перевантажений функціями, які можуть бути непотрібними для конкретного сервісного центру, що ускладнює навчання персоналу.

«Програма для сервісного центру» від розробника Універсальна Система Обліку (УСО)» [6]. «Програма для сервісного центру від УСО» має систему реєстрації та обробки клієнтських запитів, система може створювати та обробляти акти прийому/передачі, документи про технічний стан, квитанції про оплату. У програмі кожному окремому операторові можна присвоювати різноманітний рівень доступу до перегляду та внесення змін до бази даних. Також є системи пошуків за будь-якими критеріями. Сама програма та інформація знаходиться на власному ПК чи сервері СЦ.

На рисунку 1.2 представлено застосунок «Програма для сервісного центру» від УСО».

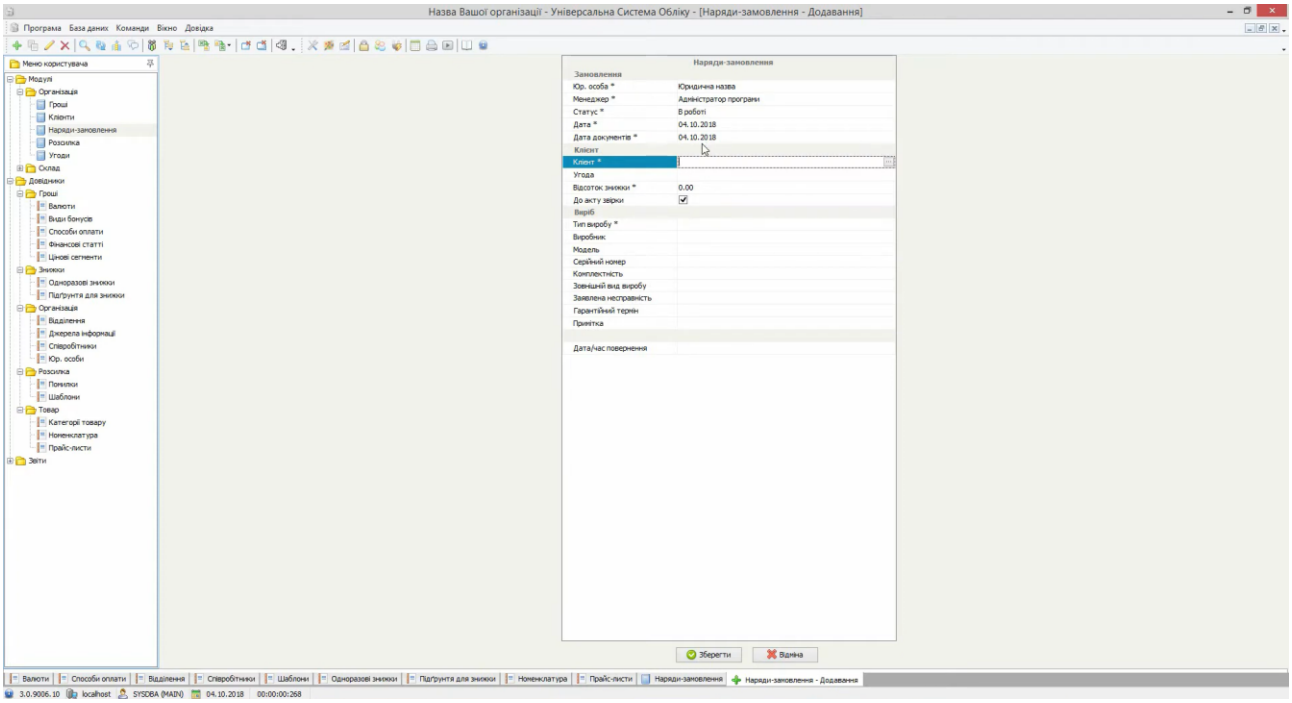


Рисунок 1.2 – Вікно створення запису «наряд-замовлення» «Програми для сервісного центру» від УСО»

Налаштування потребує технічної підтримки та навчання персоналу, що може викликати складність впровадження, а сам застосунок встановити можна тільки на ОС Windows.

«Gincore» – це комплексна хмарна ERP-система, спеціально розроблена для автоматизації сервісних центрів, ремонтних майстерень та інтернет-магазинів [7].

Однією з головних переваг даної платформи є глибоко опрацьований модуль складського обліку, який підтримує адресне зберігання, серійний облік запчастин, проведення інвентаризацій та інтеграцію з каталогами постачальників. Система охоплює повний цикл бізнес-процесів: від прийому пристрою у ремонт і маршрутизації завдань між інженерами до логістики, управління фінансами та формування деталізованої управлінської звітності.

На рисунку 1.3 представлено сторінку бухгалтерії застосунку Gincore.

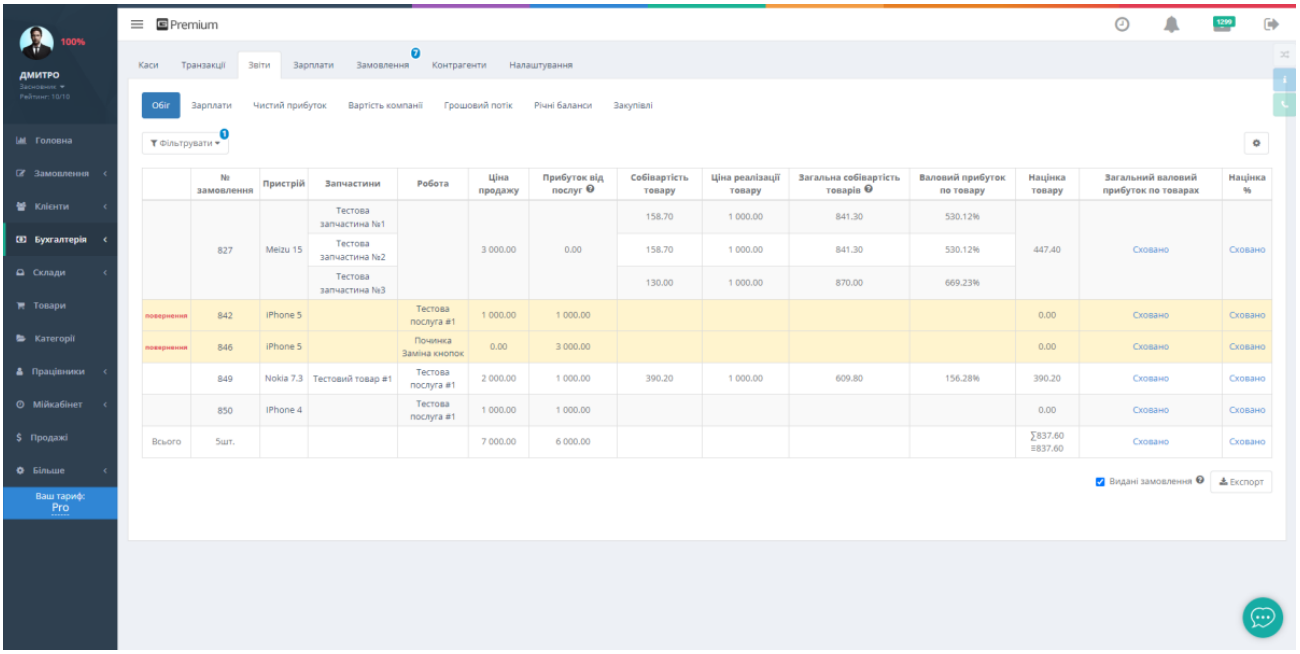


Рисунок 1.3 –Вікно бухгалтерії «Gincore»

Незважаючи на потужний функціонал, впровадження «Gincore» має низку обмежень, які важливо враховувати при виборі. Платформа працює виключно за моделлю SaaS, це формує постійне фінансове навантаження у вигляді абонентської плати, яка суттєво зростає при масштабуванні штату співробітників. Через велику кількість модулів та налаштувань система має високий поріг входження, вимагаючи значного часу та ресурсів на навчання і адаптацію персоналу.

1.3 Постановка задачі та вимоги до програмного забезпечення

Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» повинно відповідати кільком ключовим вимогам для забезпечення ефективної роботи сервісного центру та задоволення потреб клієнтів.

У межах роботи буде розроблено:

- Серверну частину програмного застосунку.

- Клієнтську частину програмного застосунку.

З програмним забезпеченням працюватимуть три основні категорії користувачів:

- Адміністратори.
- Майстри.
- Директор.

Програмне забезпечення повинно включати наступні функції:

- Автентифікацію та розмежування доступу відповідно до ролі.
- Обробку інформації про клієнтів (додавання, редагування, видалення, перегляд).
- Обробку інформації про замовлення (додавання, редагування, видалення, перегляд).
- Обробку інформації про ремонт (додавання, редагування, видалення, перегляд).
- Обробку інформації про запчастини (додавання, редагування, видалення, перегляд).
- Обробку інформації про послуги (додавання, редагування, видалення, перегляд).
- Обробку інформації про співробітників (додавання, редагування, видалення, перегляд).
- Формування звітності.
- Обробку інформації облікових записів користувачів (додавання, редагування, видалення, перегляд).

За допомогою розподілу ролей має бути обмежений доступ та відповідальність працівників до функціоналу програмного забезпечення.

Розподіл ролей працівників до функціоналу програмного забезпечення представлено у таблиці. 1.1.

Таблиця 1.1 – Розподіл ролей до функціоналу програмного забезпечення

Роль	Основні завдання	Доступ до модулів та дій
1	2	3
Адміністратор	Приймання замовлень, спілкування з клієнтами, облік інформації про клієнтів та замовлення, формування звітів.	Управління інформацією про клієнтів. Управління інформацією про замовлення. Перегляд інформації про ремонти, запчастини та послуги. Формування звітів.
Майстер	Виконання ремонтів, облік інформації ремонтів, оновлення їх статусів, фіксування використаних запчастин у ремонтах та їх облік.	Управління інформацією про ремонти. Управління інформацією про запчастини. Перегляд інформації про клієнтів, замовлення, запчастини, послуги. Формування звітів.
Директор	Управління бізнесом, персоналом, номенклатурою.	Управління інформацією про співробітників. Управління інформацією про послуги. Управління обліковими записами. Перегляд інформації про клієнтів, замовлення, ремонти, запчастини. Формування звітів.

Вимоги до складу виконуваних функцій

Програмне забезпечення повинне виконувати наступні функції:

1) Автентифікація

Вхідні: логін, пароль.

Вихідні: повідомлення про автентифікацію.

2) Дані клієнта

2.1) Додавання клієнта

Вхідні: прізвище, ім’я, по-батькові, контактні дані.

Вихідні: повідомлення про результат додавання даних про клієнта.

2.2) Редагування клієнта

Вхідні: ідентифікатор клієнта, прізвище, ім’я, по-батькові, контактні дані.

Вихідні: повідомлення про результат оновлення даних клієнта.

2.3) Видалення клієнта

Вхідні: ідентифікатор клієнта.

Вихідні: повідомлення про результат видалення даних про клієнта.

2.4) Перегляд клієнтів

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список клієнтів.

3) Дані замовлення

3.1) Створення замовлення

Вхідні: назва пристрою, виробник, модель, serial number (SNID), опис стану, опис проблеми, дата прийому, орієнтована дата видачі, дата видачі, приймальник, власник.

Вихідні: повідомлення про результат створення замовлення.

3.2) Редагування замовлення

Вхідні: ідентифікатор замовлення, назва пристрою, виробник, модель, serial number (SNID), опис стану, опис проблеми, дата прийому, орієнтована дата видачі, дата видачі, приймальник, власник.

Вихідні: повідомлення про результат оновлення даних замовлення.

3.3) Видалення замовлення

Вхідні: ідентифікатор замовлення.

Вихідні: повідомлення про результат видалення замовлення.

3.4) Перегляд замовлень

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список замовлень.

4) Дані ремонту

4.1) Додавання ремонту

Вхідні: номер замовлення, дата початку ремонту, дата кінця ремонту, майстер, послуги, запчастини, результат діагностики, вартість та статус ремонту.

Вихідні: повідомлення про результат додавання даних про ремонт.

4.2) Редагування даних про ремонт

Вхідні: ідентифікатор ремонту, номер замовлення, дата початку ремонту, дата кінця ремонту, майстер, послуги, запчастини, результат діагностики,

вартість ремонту, статус ремонту.

Вихідні: повідомлення про результат оновлення даних ремонту.

4.3) Видалення даних про ремонт

Вхідні: ідентифікатор ремонту.

Вихідні: повідомлення про результат видалення даних про ремонт.

4.4) Перегляд ремонтів

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список ремонтів.

5) Дані співробітників

5.1) Додавання співробітника

Вхідні: посада, прізвище, ім’я, по-батькові, контактний номер, кваліфікація, адреса, дата початку роботи, дата звільнення.

Вихідні: повідомлення про результат додавання даних про робітника.

5.2) Редагування співробітника

Вхідні: ідентифікатор робітника, посада, прізвище, ім’я, по-батькові, контактний номер, кваліфікація, адреса, дата початку роботи, дата звільнення.

Вихідні: повідомлення про результат оновлення даних робітника.

5.3) Видалення співробітника

Вхідні: ідентифікатор робітника, дата звільнення.

Вихідні: повідомлення про результат видалення робітника.

5.4) Перегляд робітників

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список робітників.

6) Дані послуг

6.1) Додавання послуги

Вхідні: назва послуги, опис послуги, вартість послуги.

Вихідні: повідомлення про результат додавання даних про послуги.

6.2) Редагування послуги

Вхідні: ідентифікатор послуги, назва послуги, опис послуги, вартість послуги.

Вихідні: повідомлення про результат оновлення даних послуги.

6.3) Видалення послуги

Вхідні: ідентифікатор послуги.

Вихідні: повідомлення про результат видалення послуги.

6.4) Перегляд послуг

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список послуг.

7) Дані запчастин

7.1) Додавання запчастини

Вхідні: назва запчастини, артикул запчастини, вартість покупки, вартість для ремонту, кількість.

Вихідні: повідомлення про результат додавання даних про запчастину

7.2) Редагування запчастини

Вхідні: ідентифікатор запчастини, артикул запчастини, назва запчастини, вартість покупки, вартість для ремонту, кількість.

Вихідні: повідомлення про результат оновлення даних запчастини.

7.3) Видалення запчастини

Вхідні: ідентифікатор запчастини.

Вихідні: повідомлення про результат видалення даних запчастини.

7.4) Перегляд запчастини

Вхідні: параметри пошуку та фільтрації

Вихідні: сформований список запчастин.

8) Звітність

8.1) Звіт «Акт приймання»

Вхідні: тип звіту, номер замовлення

Вихідні: Звіт «Акт приймання» з інформацією про конкретне замовлення,

клієнта та деталями про його пристрій.

8.2) Звіт «Акт видачі»

Вхідні: тип звіту, номер замовлення

Вихідні: звіт «Акт видачі» з інформацією та деталями про конкретне замовлення, клієнта та ремонту, що до нього належать.

8.3) Звіт «Фінансовий звіт»

Вхідні дані: тип звіту, початкова дата, кінцева дата.

Вихідні дані: звіт «Фінансовий звіт» з інформацією: список замовлень з виконаним статусом, з їх фінансовими даними за введений проміжок часу (оборот, витрати на запчастини, вартість послуг, прибуток).

8.4) Звіт «Використані запчастини»

Вхідні дані: тип звіту, початкова дата, кінцева дата.

Вихідні дані: звіт «Використані запчастини» з інформацією: список ремонтів за введений проміжок часу з інформацією про використання запчастин (витрачені запчастини, кількість, залишок, ціна закупки, ціна продажу, маржа).

9) Дані облікового запису

9.1) Додавання облікового запису

Вхідні: логін, пароль, роль.

Вихідні: повідомлення про результат додавання даних про обліковий запис.

9.2) Редагування облікового запису

Вхідні: ідентифікатор облікового запису, логін, пароль, роль.

Вихідні: повідомлення про результат оновлення даних облікового запису.

9.3) Видалення облікового запису

Вхідні: номер облікового запису.

Вихідні: повідомлення про результат видалення даних облікового запису.

9.4) Перегляд облікового запису

Вхідні: параметри пошуку та фільтрації

Вихідні: сформований список облікових записів.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

2.1 Аналіз вимог до програмного забезпечення «РЕМТЕСН»

2.1.1 Побудова моделі варіантів використання програмного забезпечення «РЕМТЕСН»

Модель варіантів використання є одним із основних засобів аналізу функціональних вимог до програмного забезпечення. Вона дозволяє описати взаємодію користувачів із системою, визначити основні сценарії її використання та встановити перелік функцій, які має реалізовувати програмний продукт.

Побудова моделі варіантів використання для програмного забезпечення «РемTech» дає змогу формалізувати вимоги до системи, виділити ключових акторів і визначити їхні взаємодії із програмним забезпеченням. У таблиці 2.1 представлено опис акторів.

Таблиця 2.1 – Актори програмного забезпечення для автоматизації обліку виконання робіт сервісного центру «РемTech»

Актор	Роль	Опис
1	2	3
Адміністратор	Оперативний облік	Особа яка реєструє клієнтів та замовлення, формування актів
Майстер	Виконання ремонтів	Особа яка займається обліком ремонтів та керує інформацією та звітністю про них: зміна статусів, фіксація запчастин і послуг
Директор	Управлінський облік	Особа яка керує сервісним центром. Управління персоналом і номенклатурою. Формування звітів

Варіанти використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру «РемTech» представлено у таблиці 2.2.

Таблиця 2.2 – Варіанти використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру «РемТech»

Номер варіанту використання	Назва варіанту використання	Актор	Опис
1	2	3	4
В-01	Авторизація	Адміністратор, Майстер, Директор	Введення логіну та пароля для отримання доступу до системи відповідно до ролі
В-02	Управління клієнтами	Адміністратор (CRUD), Майстер (R), Директор (R)	Перегляд, додавання, редагування та видалення записів про клієнтів
В-03	Управління замовленнями	Адміністратор (CRUD), Майстер (R), Директор (R)	Перегляд, створення, редагування та видалення замовлень; прив'язка до клієнта і приймальника
В-04	Управління ремонтами	Майстер (CRUD), Адміністратор (R), Директор (R)	Перегляд, створення, редагування та видалення ремонтів; призначення майстрів, запчастин, послуг, зміна статусу
В-05	Управління запчастинами	Майстер (CRUD), Адміністратор (R), Директор (R)	Перегляд, додавання, редагування та видалення запчастин на складі
В-06	Управління послугами	Майстер (R), Адміністратор (R), Директор (CRUD)	Перегляд, додавання, редагування та видалення позицій прайс-листу послуг
В-07	Управління співробітниками	Директор (CRUD)	Перегляд, додавання, редагування та видалення записів про персонал
В-08	Формування звітів	Адміністратор, Майстер, Директор	Формування та експорт чотирьох типів звітів (8.1–8.4) у форматах CSV, Excel, PDF
В-09	Управління обліковими записами	Директор (CRUD)	Перегляд, створення, редагування та видалення облікових записів; призначення ролей

Розподіл доступу до варіантів використання за ролями наведено у таблиці 2.3.

CRUD – це абревіатура, що розшифровується як «Create» - «Створення», «Read» - «Читання (Перегляд)», «Update» - «Оновлення» та «Delete» - «Видалення» – чотири фундаментальні операції, що формують основу взаємодії з даними [8].

Таблиця 2.3 – Розподіл доступу до варіантів використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру «РемTech»

Варіант використання	Адміністратор	Майстер	Директор
1	2	3	4
В-01 Авторизація	Доступно	Доступно	Доступно
В-02 Управління клієнтами	CRUD	R	R
В-03 Управління замовленнями	CRUD	R	R
В-04 Управління ремонтами	R	CRUD	R
В-05 Управління запчастинами	R	CRUD	R
В-06 Управління послугами	R	R	CRUD
В-07 Управління співробітниками	–	–	CRUD
В-08 Формування звітів	Доступно 8.1, 8.2	Доступно 8.4	Доступно 8.3, 8.4
В-09 Управління обліковими записами	–	–	CRUD

Діаграма варіантів використання

Діаграма варіантів використання призначена для відображення функціональних можливостей системи з точки зору користувача. Вона демонструє взаємодію акторів (користувачів або зовнішніх систем) із програмним забезпеченням через визначені сценарії використання. Така діаграма дозволяє визначити межі системи, її основні функції та ролі користувачів [9].

На основі сформульованих варіантів використання програмного забезпечення «РемTech» побудовано UML-діаграму варіантів використання, що відображає основні сценарії роботи користувачів.

На рисунку 2.1 представлено діаграму варіантів використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

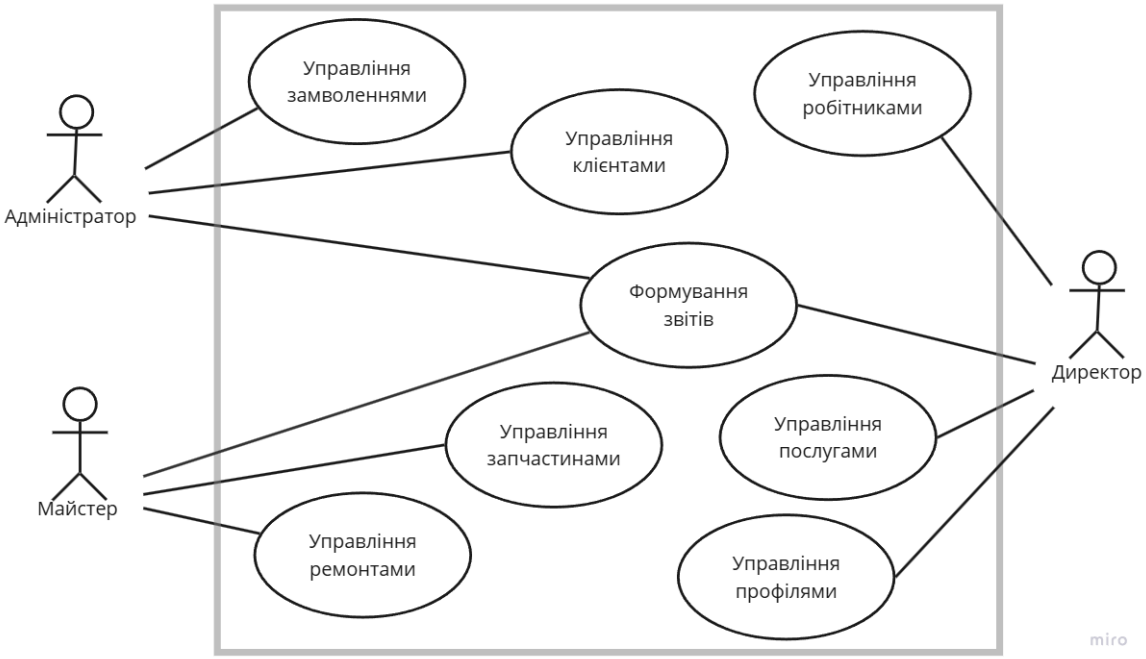


Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Авторизація є ключовим варіантом використання для отримання прав для використання інших варіантів використання та прав доступу до програмного забезпечення. Після проходження авторизації програмне забезпечення визначає роль користувача (Адміністратор, Майстер, Директор) і надає доступ лише до тих варіантів використання, що передбачені для відповідної ролі згідно з таблицею 2.3.

2.1.2 Специфікація варіантів використання програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Специфікація варіанту використання «В-02 Управління клієнтами»

Назва: В-02 Управління клієнтами

Мета: Вести базу даних клієнтів сервісного центру.

Актори: Адміністратор (CRUD), Майстер (R), Директор (R).

Передумови: Користувач авторизований.

Основний сценарій (базовий потік – Перегляд):

1. Користувач відкриває вкладку «Клієнти».

2. Таблиця відображає список клієнтів.

Розширення А1 – Додавання клієнта:

1. Користувач заповнює поля форми: прізвище, ім'я, по-батькові, телефон.

2. Натискає «Додати».

3. Застосунок перевіряє введені дані. При помилці – відображає сповіщення про помилку.

4. При успіху – застосунок надсилає запит на додавання інформації на сервер.

5. Новий запис з'являється у таблиці.

Розширення А2 – Редагування клієнта:

1. Користувач обирає запис у таблиці – форма заповнюється автоматично.

2. Редагує поля та натискає «Зберегти».

3. Застосунок надсилає запит на редагування інформації на сервер. При помилці – відображає сповіщення про помилку.

4. При успіху – застосунок надсилає запит на редагування інформації на сервер.

5. Запис у таблиці оновлюється.

Розширення А3 – Видалення клієнта:

1. Користувач обирає запис і натискає «Видалити».

2. Застосунок надсилає запит на видалення інформації на сервер.

3. Запис зникає з таблиці.

Постумови: Зміни в базі даних серверу і відображення актуальних даних у всіх клієнтських застосунках. При існуючому записі – доступне для вибору при створенні замовлення (В-03).

Специфікація варіанту використання «В-03 Управління замовленнями»

Назва: В-03 Управління замовленнями

Мета: Реєструвати та вести облік замовлень на ремонт.

Актори: Адміністратор (CRUD), Майстер (R), Директор (R).

Передумови: Авторизований. Існує щонайменше запис про одного клієнта та одного співробітника.

Основний сценарій (базовий потік – Перегляд):

1. Користувач відкриває вкладку «Замовлення».

2. Застосунок відображає таблицю зі всіма замовленнями з полями: пристрій, виробник, модель, SNID, клієнт, приймальник, дати.

Розширення А1 – Додавання замовлення:

1. Користувач заповнює форму: тип пристрою, виробник, модель, SNID, опис стану, опис проблеми, дата прийому, орієнтовна дата видачі.

2. Обирає клієнта та приймальника.

3. Натискає «Додати».

4. Застосунок перевіряє введені дані. При помилці – відображає сповіщення про помилку.

5. При успіху – застосунок надсилає запит на додавання інформації на сервер.

6. Новий запис з'являється у таблиці.

Розширення А2 – Редагування замовлення: аналогічно В-02 (А2).

Розширення А3 – Видалення замовлення: аналогічно В-03 (А3).

Постумови: Замовлення збережено. Доступне для вибору при створенні ремонту (УВ-04).

Специфікація варіанту використання «В-08 Формування звітів»

Назва: В-08 Формування звітів

Мета: Отримати структуровані звіти про діяльність сервісного центру та експортувати їх.

Актори: Адміністратор, Майстер, Директор.

Передумови: Авторизований. Існують необхідні дані (замовлення / ремонти).

Основний сценарій:

Користувач відкриває вкладку «Звітність».

Користувач обирає тип звіту:

8.1 – Акт приймання пристрою: вводить номер замовлення -> застосунок відображає дані клієнта, пристрою, стану, проблеми, підпис.

8.2 – Акт видачі (Квитанція): вводить номер замовлення -> застосунок відображає дані клієнта, пристрою, виконаних робіт і суму до сплати.

8.3 – Фінансовий звіт: вводить діапазон дат -> застосунок фільтрує замовлення, дата видачі яких відповідає діапазону та зі статусом ремонту «Виконано» і відображає таблицю: замовлення, оборот, витрати на запчастини, послуги, прибуток.

8.4 – Використані запчастини: вводить діапазон дат -> застосунок агрегує використання запчастин ремонтів дата видачі яких відповідає діапазону і відображає: назву, артикул, кількість, залишок на складі, суму.

Натискає «Сформувати звіт».

1. Застосунок відображає результат і підсумковий рядок.

2. За бажанням, користувач натискає збережує файл.

Альтернативний сценарій А1 – Немає даних / Некоректні вхідні дані:

Застосунок відображає порожню таблицю або повідомлення про відсутність записів за вхідними даними.

Постумови: Звіт сформовано та відображено у застосунку, файл збережено на комп'ютері у обраному форматі (за умови збереження).

2.2 Архітектура програмного забезпечення для автоматизації обліку виконання робіт сервісного центру с ремонту комп’ютерної техніки «РемTech»

Використання клієнт-серверної архітектури є доцільним і обґрунтованим з урахуванням вимог до розробки програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech». Такий підхід дозволяє розділити застосунок на незалежні компоненти, забезпечити масштабованість, централізоване зберігання даних і можливість одночасної роботи кількох клієнтів.

Клієнтська частина програмного забезпечення «РемTech» реалізується з використанням архітектурного патерну MVVM (Model-View-ViewModel), основною метою якого є чітке розмежування логіки інтерфейсу користувача та бізнес-логіки застосунку [10].

Структура MVVM включає три основні компоненти:

Model (Модель) – рівень даних і бізнес-логіки. Містить сутності предметної області, класи доступу до даних, а також об’єкти передачі даних (DTO), які надходять із серверної частини. Модель не має залежностей від інтерфейсу користувача.

View (Представлення) – візуальна частина застосунку, яка відповідає за відображення даних користувачеві. У межах патерну MVVM представлення не повинно містити бізнес-логіки; допускається лише ініціалізація компонентів інтерфейсу.

ViewModel (Модель представлення) – проміжний рівень між Model і View, який забезпечує перетворення даних у формат, зручний для відображення, а також обробку дій користувача. ViewModel містить стан інтерфейсу (наприклад, доступність кнопок) і команди, що виконуються у відповідь на події. Взаємодія між View і ViewModel здійснюється через механізм прив’язки даних (Data Binding), без прямої залежності.

Уніфікована мова моделювання UML (Unified Modeling Language) є стандартом для візуалізації, проєктування та документування програмних систем. Вона дозволяє описувати як структурні, так і поведінкові аспекти системи за допомогою набору графічних діаграм, що спрощує розуміння архітектури та взаємодії компонентів програмного забезпечення [9].

На рисунку 2.2 представлено діаграму взаємодії патерну MVVM.

Рисунок 2.2 – Діаграма взаємодії патерну MVVM

Серверна частина програмного забезпечення «РемTech» реалізується відповідно до принципів багатошарової архітектури (Layered Architecture) [11], яка є поширеним підходом при розробці корпоративних застосунків. Така архітектура забезпечує розподіл відповідальності між окремими компонентами програмного застосунку та підвищує її модульність, гнучкість і підтримуваність.

У межах цього підходу програмне забезпечення поділяється на логічні рівні, кожен із яких виконує окрему функцію та взаємодіє лише з сусідніми рівнями. Наприклад, контролери взаємодіють із сервісами, а сервіси – з рівнем доступу до даних.

Структура багатошарової архітектурим (N-Layer) включає такі основні компоненти:

Presentation / Controller Layer (Рівень контролерів) – обробляє HTTP-запити (GET, POST, PUT, DELETE), що надходять від клієнтської частини. Дані приймаються у форматі JSON, перетворюються у DTO та передаються до сервісного рівня.

Service Layer (Сервісний рівень) – містить бізнес-логіку застосунку. На цьому рівні реалізуються правила обробки даних, перевірка коректності операцій

та координація взаємодії між компонентами.

Data Access Layer / Repository (Рівень доступу до даних) – відповідає за взаємодію з базою даних, виконання запитів та отримання або збереження даних.

На рисунку 2.3 представлено діаграму взаємодії патерну Layered Architecture.

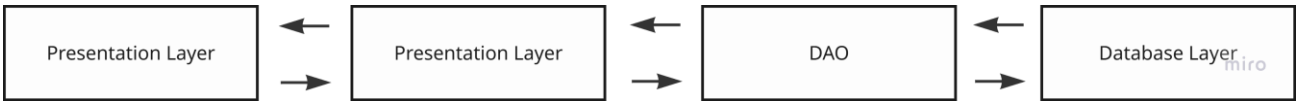


Рисунок 2.3 – Діаграма взаємодії патерну Layered Architecture

Для представлення структури даних у системі використовуються сутності (Entity), які відображають таблиці бази даних та описують об’єкти предметної області і зв’язки між ними.

Обмін даними між серверною та клієнтською частинами здійснюється за допомогою DTO (Data Transfer Objects), які дозволяють передавати лише необхідну інформацію без прямої залежності від внутрішньої структури бази даних.

Діаграма компонентів використовується для представлення загальної структури програмного забезпечення, зокрема його складових частин та зв’язків між ними. Вона відображає архітектуру системи на високому рівні абстракції, показуючи, як окремі компоненти взаємодіють між собою [9].

На рисунку 2.4 наведено діаграму компонентів програмного забезпечення «РемTech», яка ілюструє загальну архітектуру та її основні складові.

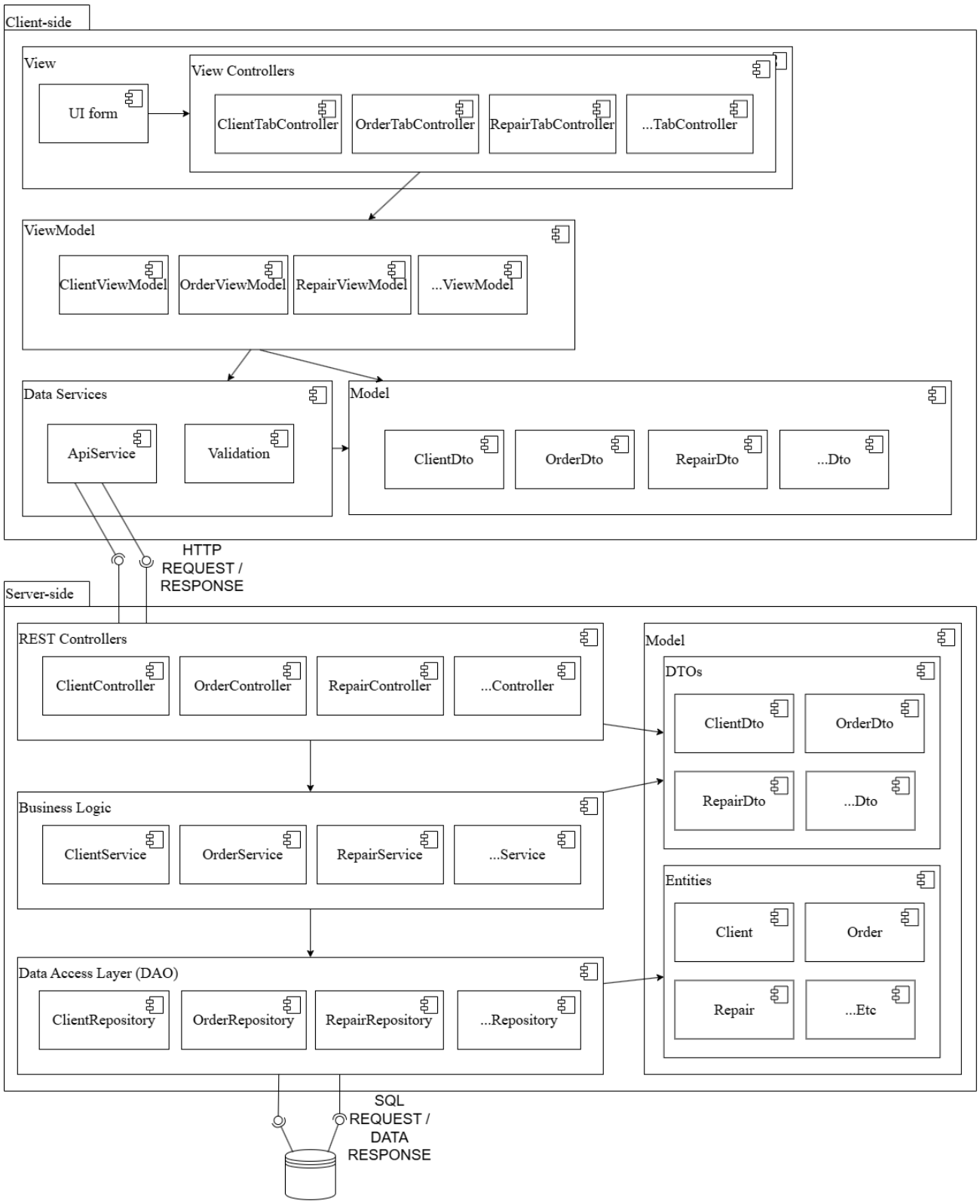


Рисунок 2.4 – Діаграма компонентів програмного забезпечення «РемTech»

Загальну структуру програмного забезпечення доцільно представляти за допомогою діаграми компонентів, яка є складовою уніфікованої мови моделювання UML. Вона належить до структурних діаграм і використовується для відображення організації системи та взаємозв’язків між її компонентами

Оптимальним інструментом для представлення фізичної структури системи є діаграма розгортання UML. Вона ілюструє, як саме програмне забезпечення розподіляється між апаратними потужностями, описуючи топологію вузлів, фізичні артефакти та їхню організацію [9].

У межах програмного забезпечення «РемTech» діаграма розгортання відображає взаємодію клієнтської та серверної частин, розміщених на різних вузлах, а також використання мережевих протоколів для обміну даними між ними.

На рисунку 2.5 представлено діаграму розгортання програмного забезпечення «РемTech»

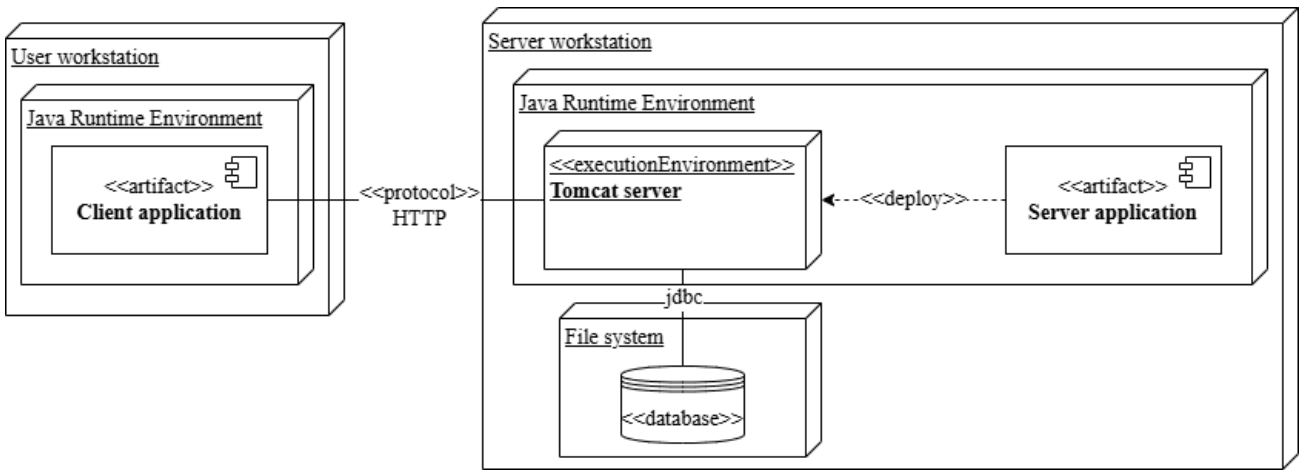


Рисунок 2.5 – Діаграма розгортання програмного забезпечення «РемTech»

Таким чином, обрана клієнт-серверна архітектура забезпечує централізоване зберігання даних, масштабованість системи та можливість одночасної роботи кількох клієнтських вузлів без дублювання бази даних.

2.3 Проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»

2.3.1 Ескізний проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»

Діаграма діяльності

Діаграма діяльності використовується для моделювання сценаріїв роботи користувача із системою, опису бізнес-процесів та деталізації логіки виконання окремих функцій. Вона є корисною на етапі проєктування програмного забезпечення, оскільки дозволяє виявити можливі логічні помилки та оптимізувати процеси ще до їх реалізації [9].

На рисунку 2.6 представлено узагальнену діаграму діяльності для сценаріїв додавання інформації сутності.

Рисунок 2.6 – Узагальнена діаграма діяльності сценаріїв додавання інформації про сутності

Оскільки операції створення, перегляду, редагування та видалення (CRUD) для різних сутностей системи реалізуються за аналогічним сценарієм, доцільно представити узагальнену діаграму діяльності без деталізації для кожної окремої сутності.

Концептуальна модель даних

Концептуальна діаграма використовується для узагальненого представлення предметної області системи. Вона відображає основні сутності та зв’язки між ними без деталізації реалізації. Така діаграма дозволяє сформувати цілісне уявлення про структуру системи та є основою для подальшого проєктування [9].

На рисунку 2.7 представлена концептуальна модель даних у вигляді діаграми «сутність-зв'язок».

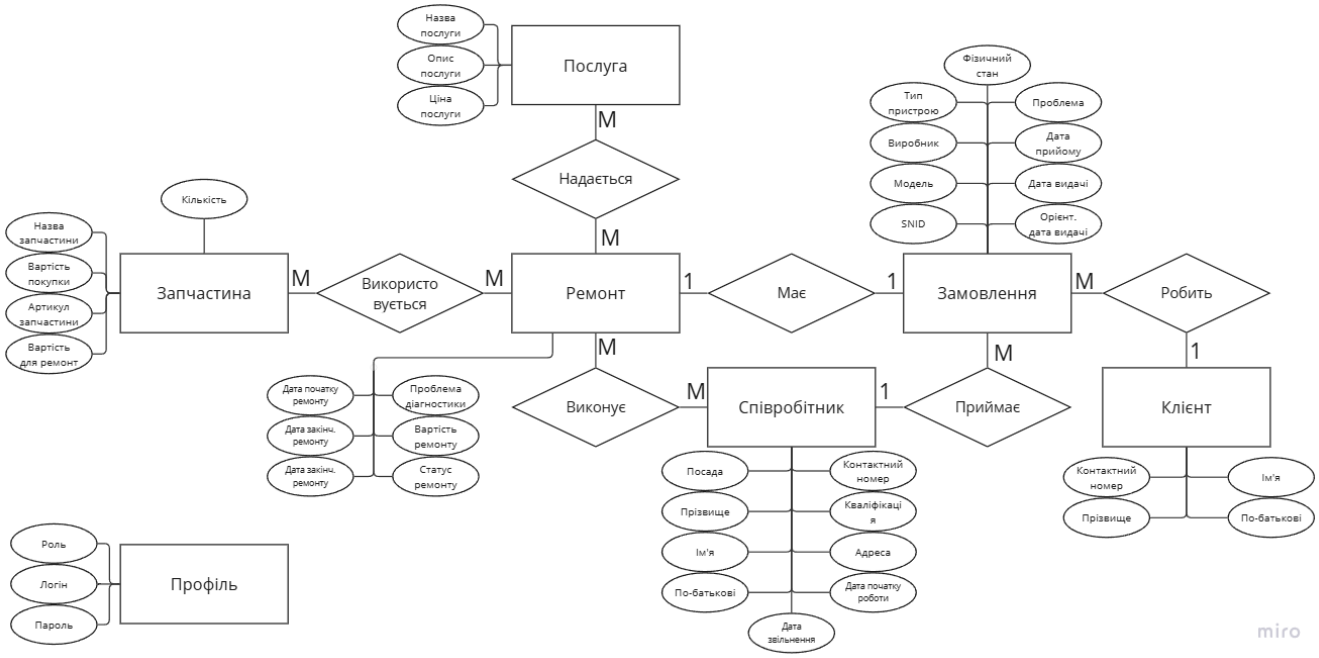


Рисунок 2.7 – Концептуальна модель даних

Прототип інтерфейсу користувача

Прототип відображає структуру елементів, їх взаємне розташування та основні сценарії взаємодії користувача із системою: перемикання таблиць відповідними вкладками; панель із формою. для заповнення інформації, панель управління записами та повідомлення без додаткових вікон біля таблиці, для спрощення та прискорення взаємодії користувача із застосунком.

Прототип екрану «Ремонт» застосунку «РемTech» представлено на рисунку 2.8.

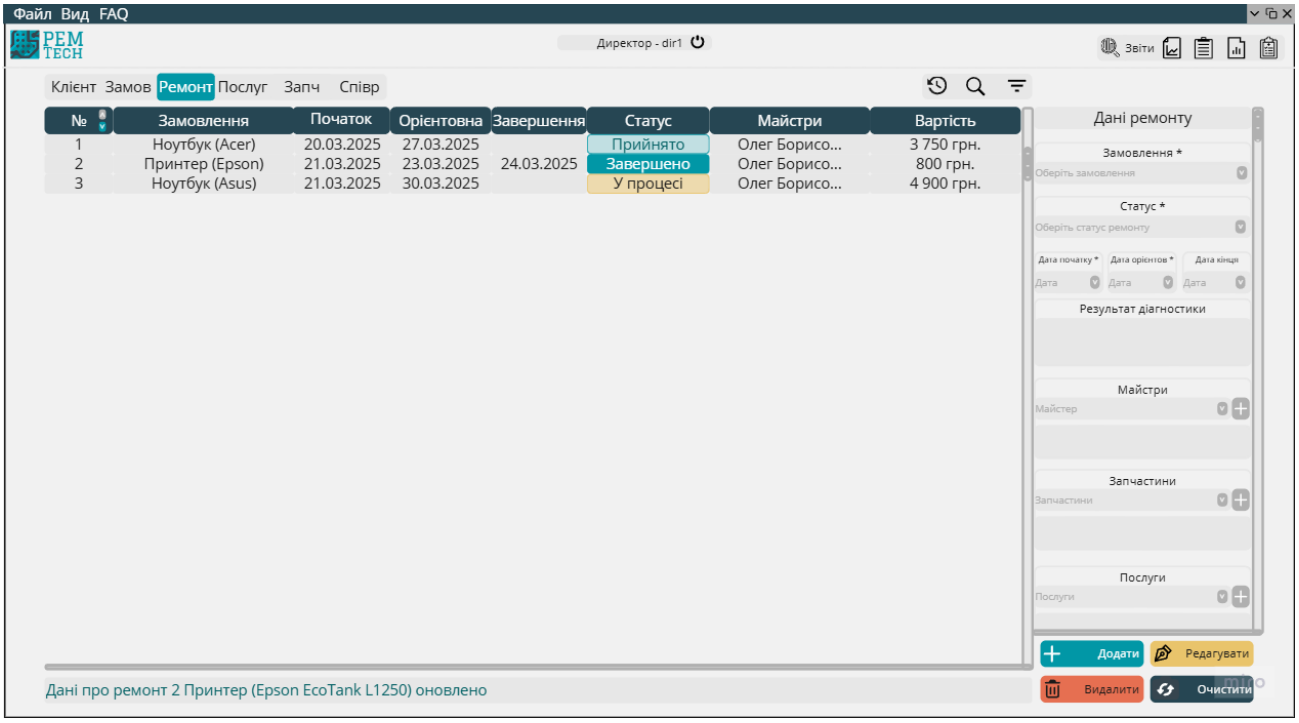


Рисунок 2.8 – Прототип екрану «Ремонт» застосунку «РемTech»

Прототип екрану авторизації застосунку «РемTech» представлено на рисунку 2.9.

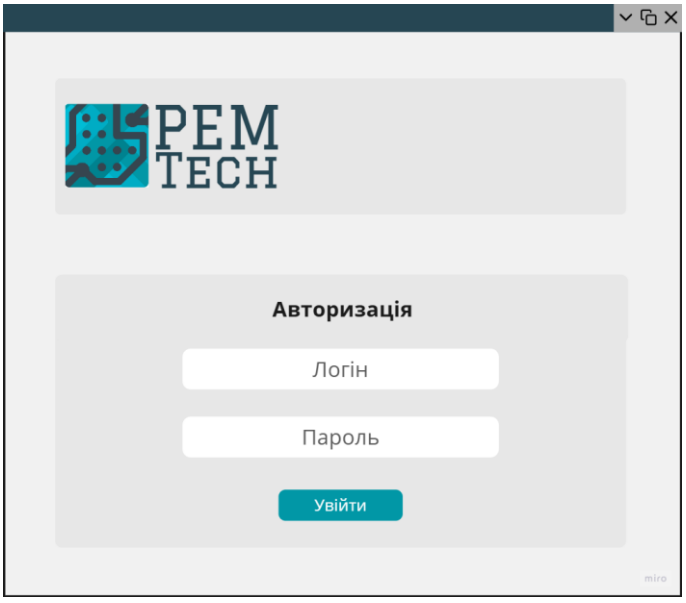


Рисунок 2.9 – Прототип екрану авторизації застосунку «РемTech»

Розроблені прототипи інтерфейсу дозволили на ранньому етапі узгодити структуру екранів та логіку взаємодії користувача із системою, що стало основою для подальшої реалізації інтерфейсу у JavaFX.

2.3.2 Технічний проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»

Діаграма класів є основною структурною діаграмою UML, яка відображає статичну структуру системи. Вона включає класи, їх атрибути, методи та зв’язки між ними (асоціації, наслідування, залежності). Дана діаграма використовується для проєктування структури програмного забезпечення [9].

На рисунку 2.10 представлена діаграма класів клієнтської частини, яка стосується сутності клієнта.

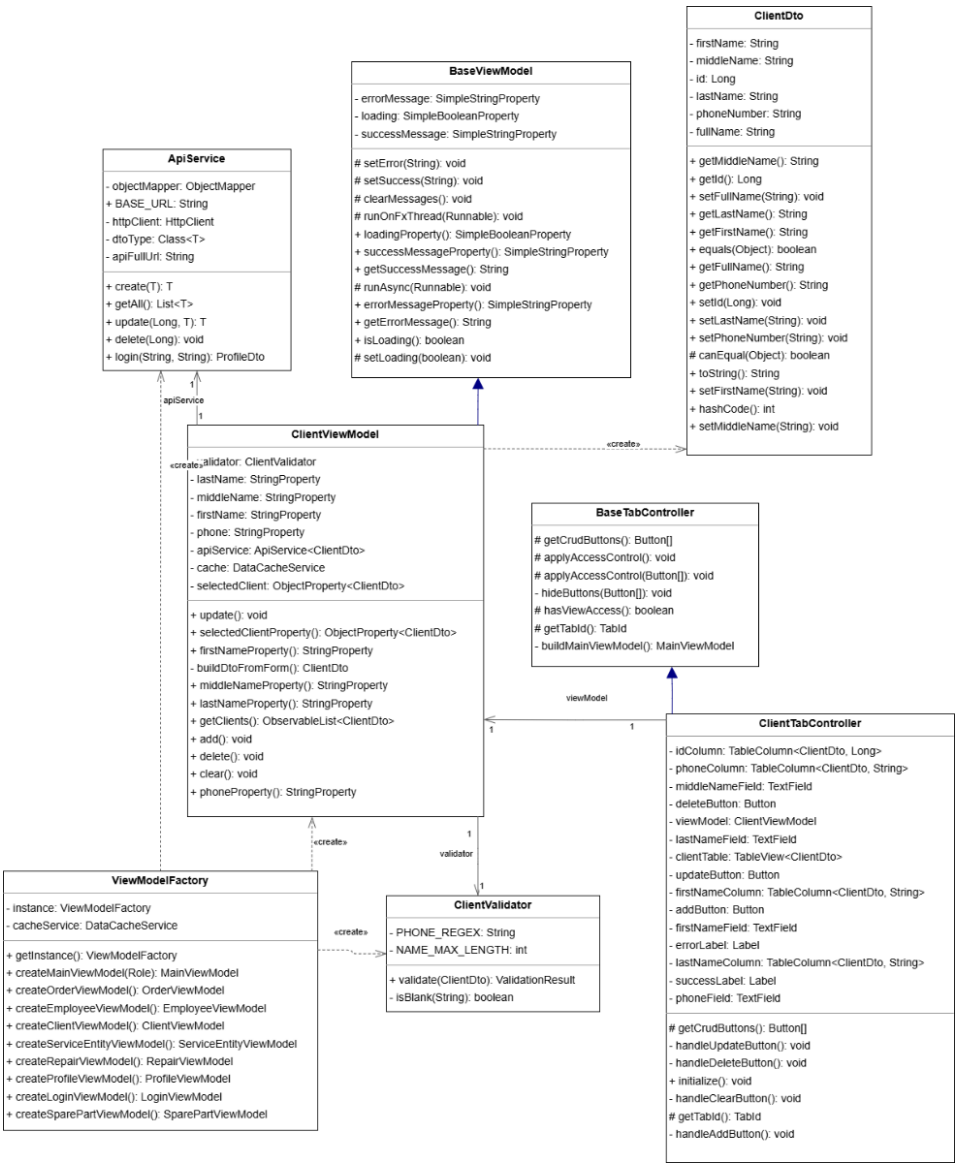


Рисунок 2.10 – Діаграма класів клієнтської частини застосунку «РемTech»

На рисунку 2.11 представлена діаграма класів серверної частини, яка стосується сутності клієнта.

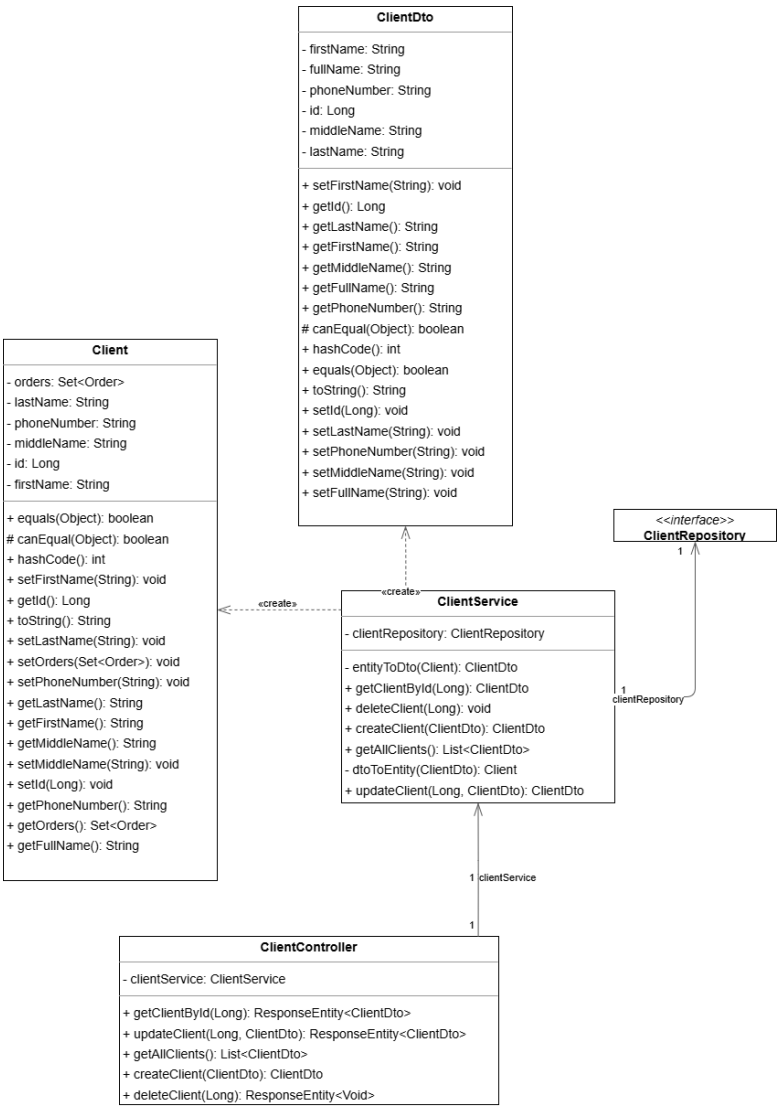


Рисунок 2.11 – Діаграма класів серверної частини застосунку «РемTech»

Специфікації класів програмного забезпечення для автоматизації обліку роботи сервісного центру «РемTech»

Ці класи забезпечують взаємодію користувача з даними та логіку відображення.

Клас ClientTabController (View)

Роль: Контролер графічного інтерфейсу (вкладка клієнтів).

Функції: Ініціалізує компоненти UI (таблицю, текстові поля), налаштовує двосторонній біндінг даних із ClientViewModel, обробляє події натискання кнопок (додати, оновити, видалити).

Клас ClientViewModel (ViewModel)

Роль: Посередник між інтерфейсом та бізнес-логікою.

Функції: Зберігає стан полів вводу та вибраного клієнта за допомогою властивостей (lastNameProperty, phoneProperty тощо). Виконує асинхронні операції додавання, редагування та видалення клієнтів через ApiService, а також оновлення даних на клієнті.

Клас ClientDto (Data Transfer Object)

Роль: Об'єкт для передачі даних клієнта між шарами додатка.

Функції: Містить поля: id, lastName, firstName, middleName, fullName та phoneNumber.

Клас ClientValidator (Validation Logic)

Роль: Перевірка коректності введених даних клієнта.

Функції: Перевіряє обов'язковість заповнення прізвища та імені, а також валідує формат номера телефону за допомогою регулярного виразу.

Клас ApiService<ClientDto> (Network Service)

Роль: Універсальний сервіс для виконання HTTP-запитів до сервера.

Функції: Реалізує методи getAll(), create(), update() та delete() для взаємодії з REST API за адресою /api/clients.

Серверна частина

Ці класи забезпечують обробку запитів, бізнес-логіку на стороні сервера та роботу з базою даних.

Клас ClientController (REST Controller)

Роль: Точка входу для HTTP-запитів (API Endpoint).

Функції: Обробляє запити за шляхом /api/clients. Використовує анотації Spring (@PostMapping, @GetMapping тощо) для мапінгу операцій створення, отримання, оновлення та видалення клієнтів.

Клас ClientService (Business Service)

Роль: Рівень бізнес-логіки сервера.

Функції: Виконує транзакційні операції над даними. Реалізує логіку перетворення (mapping) між ClientDto та сутністю Client, забезпечує пошук клієнта за ID та оновлення його властивостей.

Клас ClientRepository (Data Access Layer)

Роль: Інтерфейс для прямої взаємодії з базою даних.

Функції: Успадковує JpaRepository, що надає стандартні методи для роботи з таблицею clients (пошук, збереження, видалення) без необхідності написання SQL-запитів вручну.

Клас Client (Persistence Entity)

Роль: Модель даних для відображення в БД.

Функції: Описує структуру таблиці clients, включаючи поля id, lastName, firstName, middleName, phoneNumber та зв'язок «один-до-багатьох» із замовленнями (orders).

Клас ClientDto (Server DTO)

Роль: Об'єкт передачі даних на стороні сервера.

Функції: Містить анотації валідації (@NotBlank, @Pattern), які Spring автоматично перевіряє при отриманні JSON-запиту від клієнта.

Діаграма взаємодії показує взаємодію, що складається з набору об'єктів та їх зв'язків, включаючи повідомлення, які можуть бути відправлені між ними [9].

Діаграма послідовності прецеденту «Додавання нового клієнту» зображена на рисунку 2.12.

Логічна модель даних визначає структуру даних на концептуальному рівні, фокусуючись на вмісті та взаємозв'язках даних та бізнес логіки без урахування технічних деталей їх зберігання. Вона включає основні сутності їх атрибути (поля) та відносини між ними (наприклад, один клієнт може мати декілька замовлень). Така модель використовується для проектування структури бази даних та служить мостом між аналізом вимог та технічною реалізацією [12].

Логічна модель даних застосунку «РемTech» представлена на рисунку 2.13.

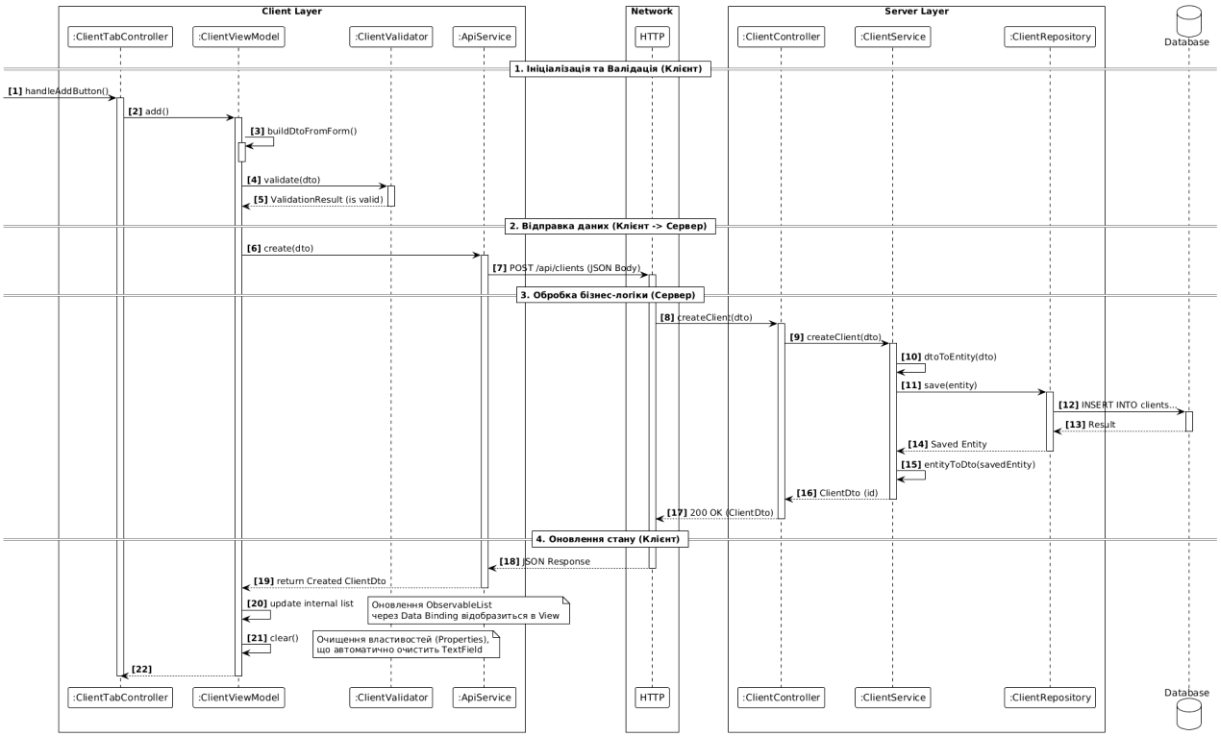


Рисунок 2.12 – Діаграма послідовності прецеденту «Додавання нового клієнта»

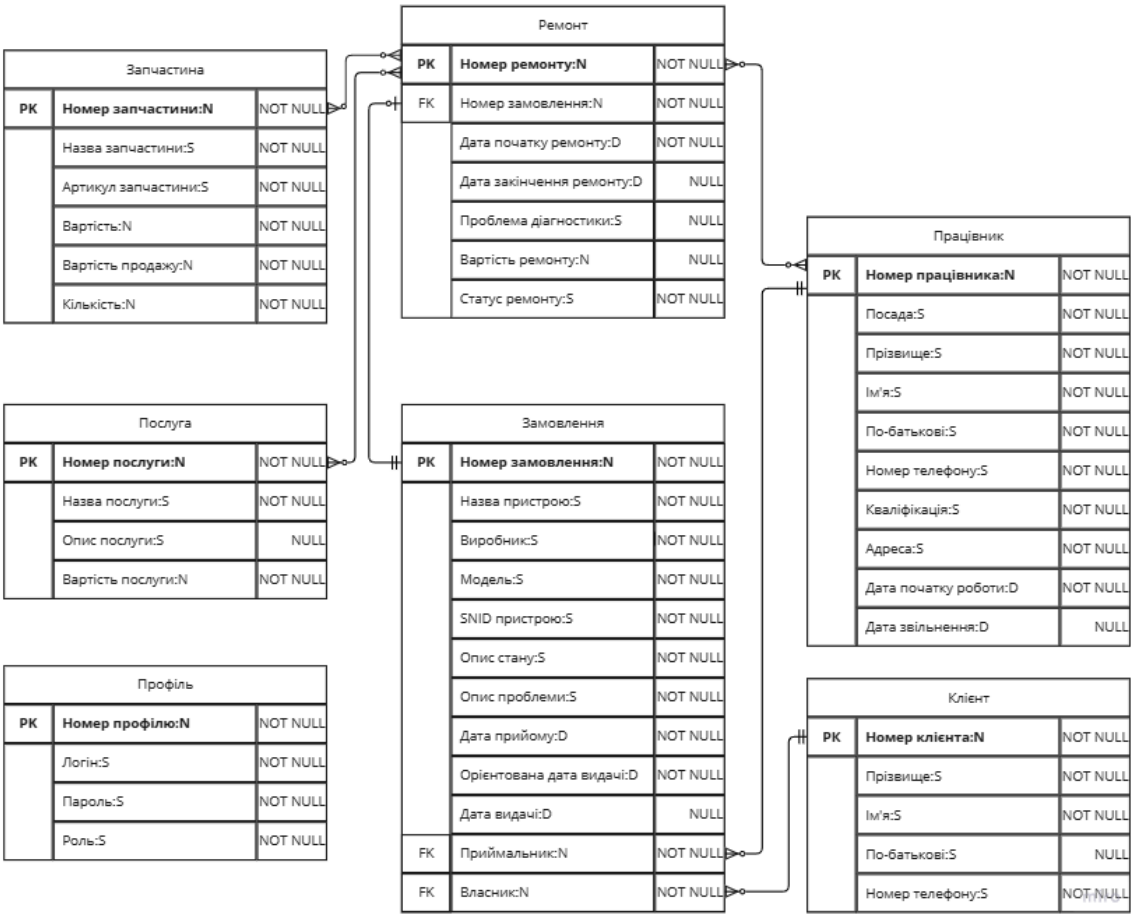


Рисунок 2.13 – Логічна модель даних застосунку «РемTech»

Побудована логічна модель даних відображає всі ключові сутності предметної області та зв'язки між ними, що стане підґрунтям для розробки фізичної моделі та реалізації бази даних.

2.3.3 Робочий проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech»

У даному розділі було зроблено та обґрунтовано вибір засобів розробки, розроблено відповідне програмне забезпечення, проведено тестування та випробовування.

Обґрунтування вибору засобів розробки

Основою для розробки програмного застосунку «РемTech» обрано мову Java. Вибір зумовлений її високою надійністю, суворою типізацією та розвиненою екосистемою для корпоративних рішень [13]. Використання віртуальної машини Java JVM забезпечує кросплатформність рішення, дозволяючи інтегрувати програмне забезпечення у наявну IT-інфраструктуру сервісного центру незалежно від встановлених операційних систем [14]. Такий підхід дозволить ще на етапі розробки мінімізувати логічні помилки в модулях, що відповідають за фінансові розрахунки та обробку даних.

Для зберігання інформації існують реляційні СУБД, наприклад PostgreSQL, MySQL, MariaDB. Вони гарантують дотримання принципів ACID (атомність, узгодженість, ізольованість, довговічність) [15], що є обов'язковим для надійної автоматизації обліку даних та складських залишків. Поряд із серверними СУБД також існують вбудовані рішення, зокрема SQLite та H2, які не потребують окремого розгортання серверу баз даних і можуть працювати безпосередньо у складі застосунку. Це значно спрощує процес встановлення та використання програмного забезпечення, оскільки відсутня необхідність додаткового адміністрування та налаштування середовища.

У проєкті було обрано СУБД H2 як оптимальний компроміс між функціональністю повноцінних серверних рішень і зручністю використання вбудованих баз даних.

Для реалізації серверної частини програмного забезпечення використано фреймворк Spring Boot, який є сучасним інструментом для створення веб-застосунків на платформі Java. Він забезпечує спрощену конфігурацію, вбудований сервер додатків та підтримку розробки REST сервісів, що дозволяє значно прискорити процес створення серверної логіки та організацію взаємодії з клієнтською частиною [16].

Spring Data JPA (Hibernate): Використовується як рівень абстракції над базою даних. Це дозволяє працювати з записами БД як з Java-об'єктами (Entities),

автоматизуючи генерацію SQL-запитів та забезпечуючи цілісність даних [17].

Jakarta Validation: Для гарантування коректності вхідних даних на рівні сервера застосовано декларативну валідацію (анотації @NotBlank, @Size, @Pattern), що запобігає збереженню невалідних записів у системі [18].

З огляду на потребу персоналу в інтенсивній роботі з даними, для створення клієнтської частини обрано бібліотеку JavaFX [19]. Вона дозволяє реалізувати повнофункціональний десктопний застосунок, де дизайн інтерфейсу, може бути описаний за допомогою FXML та CSS-стилів, та буде чітко відокремлений від програмного коду.

Розробка фізичної моделі бази даних

Фізична модель даних – це конкретне представлення даних з урахуванням особливостей системи управління базами даних (СКБД), де буде побудована база даних. Вона описує таблиці, їх атрибути, типи даних, індекси, зв'язок між таблицями, і забезпечує цілісність даних.

Фізична модель включає реалізацію логічної структури з урахуванням продуктивності, безпеки та інших технічних аспектів [20].

Фізична модель даних для програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech» представлена на рисунку 2.14.

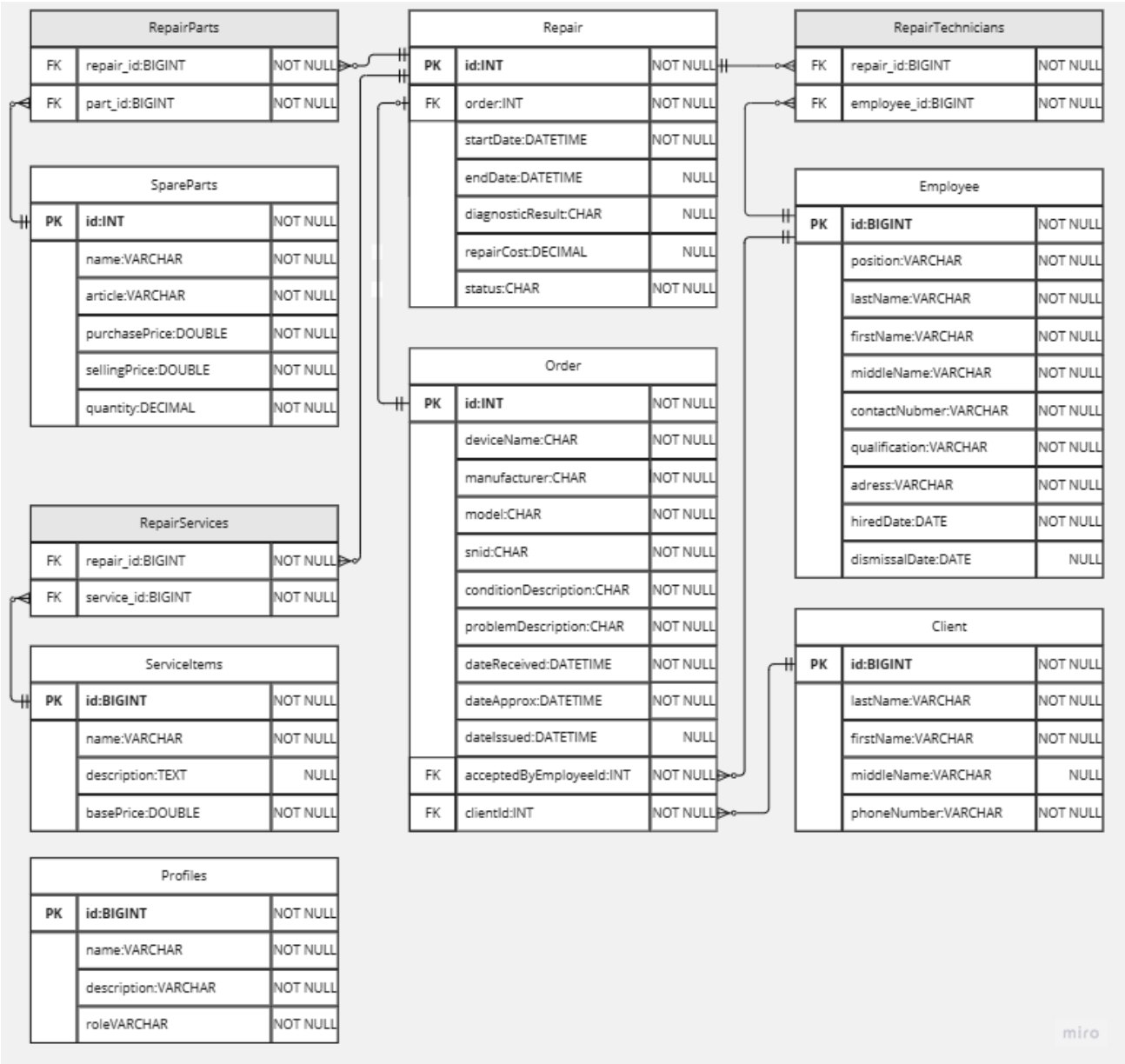


Рисунок 2.14 – Фізична модель даних застосунку «РемTech»

Розроблена фізична модель даних повністю відповідає функціональним вимогам, визначеним у технічному завданні, та забезпечує зберігання всіх необхідних сутностей предметної області сервісного центру.

Реалізація програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»

Фрагменти коду реалізації створення клієнта надано нижче для серверної

частини застосунку.

ClientController.java (Server-side):

```
@RestController
@RequestMapping("/api/clients")
public class ClientController {
    @Autowired
    private ClientService clientService;
    @PostMapping
    public ClientDto createClient(@Valid @RequestBody ClientDto
clientDto) {
        return clientService.createClient(clientDto);
    }
    @GetMapping
    public List<ClientDto> getAllClients() {
        return clientService.getAllClients();
    }
    @GetMapping("/{id}")
    public ResponseEntity<ClientDto> getClientById(@PathVariable
Long id) {
        try {
            return
ResponseEntity.ok(clientService.getClientById(id));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }
    @PutMapping("/{id}")
    public ResponseEntity<ClientDto> updateClient(@PathVariable Long
id,
                                                  @Valid
@RequestBody ClientDto clientDto) {
        try {
            return ResponseEntity.ok(clientService.updateClient(id,
clientDto));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteClient(@PathVariable Long id)
{
        clientService.deleteClient(id);
        return ResponseEntity.noContent().build();
    }
}
```

ClientService.java (Server-side):

```
@Service
```

```
public class ClientService {
    @Autowired
    private ClientRepository clientRepository;
    @Transactional
    public ClientDto createClient(ClientDto dto) {
        Client client = dtoToEntity(dto);
        Client savedClient = clientRepository.save(client);
        return entityToDto(savedClient);
    }
    @Transactional(readOnly = true)
    public List<ClientDto> getAllClients() {
        return clientRepository.findAll()
                .stream()
                .map(this::entityToDto)
                .collect(Collectors.toList());
    }
    @Transactional(readOnly = true)
    public ClientDto getClientById(Long id) {
        Client client = clientRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Клієнта з
ID " + id + " не знайдено"));
        return entityToDto(client);
    }
    @Transactional
    public ClientDto updateClient(Long id, ClientDto dto) {
        Client client = clientRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Клієнта
для оновлення не знайдено"));
        client.setFirstName(dto.getFirstName());
        client.setLastName(dto.getLastName());
        client.setMiddleName(dto.getMiddleName());
        client.setPhoneNumber(dto.getPhoneNumber());

        return entityToDto(clientRepository.save(client));
    }
    @Transactional
    public void deleteClient(Long id) {
        if (!clientRepository.existsById(id)) {
            throw new RuntimeException("Неможливо видалити: клієнта
не знайдено");
        }
        clientRepository.deleteById(id);
    }
}
```

ClientDto.java (Server-side):

```
@Data
public class ClientDto implements Serializable {
    private Long id;
```

```
    @NotBlank(message = "Прізвище обов'язкове")
    @Size(max = 50)
    private String lastName;
    @NotBlank(message = "Ім'я обов'язкове")
    @Size(max = 50)
    private String firstName;
    @Size(max = 50)
    private String middleName;
    private String fullName; // для зручності відображення клієнта
    @NotBlank(message = "Номер телефону обов'язковий")
    @Pattern(regexp = "^\\+?[0-9]{10,13}$", message = "Некоректний
формат телефону")
    private String phoneNumber;
}
```

ClientRepository.java (Server-side):

```
public interface ClientRepository extends JpaRepository<Client,
Long> { }
```

Розробка програмного забезпечення виконана на мові Java з використанням фреймворку Spring Boot з дотриманням принципів об'єктно-орієнтованого програмування (ООР), патернів SOLID [21] та клієнт-серверної архітектури із використанням патернів N-Layer для серверної частини застосунку та MVVM для клієнтської частини застосунку. Нижче наведено пояснення ключових фрагментів коду на прикладі сутності «Клієнт» – вона повністю ілюструє всі шари системи і є репрезентативною для решти модулів (замовлення, ремонти, запчастини тощо).

ClientController є REST-контролером для обробки HTTP-запитів щодо клієнтів. Анотації @RestController і @RequestMapping("/api/clients") визначають його як компонент Spring із базовим маршрутом /api/clients. Методи контролера реалізують CRUD-операції: створення (@PostMapping), отримання списку (@GetMapping), пошук за ID (@GetMapping("/{id}")), оновлення (@PutMapping("/{id}")) та видалення (@DeleteMapping("/{id}")). Дані передаються через ClientDto, а валідація виконується за допомогою @Valid.

Client є JPA-сутністю, що відображається на таблицю clients. Анотації Lombok автоматично генерують стандартні методи, а правила валідації

(@NotBlank, @Size, @Pattern) забезпечують коректність даних. Поле phoneNumber має обмеження унікальності. Зв’язок @OneToMany реалізує відношення «один клієнт – багато замовлень», використовуючи каскадні операції та відкладене завантаження (LAZY). Метод getFullName() формує повне ім’я клієнта без збереження його в БД.

ClientService містить бізнес-логіку роботи з клієнтами та зареєстрований як Spring-сервіс (@Service). Усі операції виконуються в межах транзакцій (@Transactional), а методи читання використовують режим readOnly = true для оптимізації. Сервіс також виконує перетворення між ClientDto та сутністю Client, забезпечуючи розділення API та доменної моделі.

ClientRepository успадковує JpaRepository<Client, Long>, завдяки чому Spring Data JPA автоматично надає стандартні методи роботи з базою даних без написання SQL-коду.

Повний текст програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» представлено у Додатку Б.

Тестування програмного забезпечення «РемTech» виконано методом класів еквівалентності (Equivalence Class Partitioning) [22] – одним із базових методів тестування «чорної скриньки». Суть методу полягає у розбитті множини можливих вхідних значень кожного поля на класи, у межах яких застосунок поводиться однаково. Для кожного класу формується мінімум один тест. Виділяють два види класів:

Правильні класи еквівалентності (ПКЕ) – значення, які застосунок повинен обробити без помилки.

Неправильні класи еквівалентності (НКЕ) – значення, які застосунок повинен відхилити та повідомити про помилку.

Тест 1. Тестування варіанту використання «Автентифікація» (В-01).

Виділені класи еквівалентності для функції «Автентифікація» наведено у

таблиці 2.4.

Таблиця 2.4 – Виділення класів еквівалентності функції «Автентифікація» (В-01)

№	Поле	Умова	Клас	Тип
1	Логін	Рядок, що відповідає наявному обліковому запису	КЕ1	ПКЕ
2	Логін	Порожній рядок	КЕ2	НКЕ
3	Логін	Рядок, якого немає в системі	КЕ3	НКЕ
4	Пароль	Рядок, що збігається з паролем обраного облікового запису	КЕ4	ПКЕ
5	Пароль	Порожній рядок	КЕ5	НКЕ
6	Пароль	Рядок, що не збігається з паролем	КЕ6	НКЕ

Тести для правильних класів еквівалентності функції «Автентифікація» наведено у таблиці 2.5.

Таблиця 2.5 – Тести для правильних класів еквівалентності функції «Автентифікація» (В-01)

ID тесту	Поле «Логін»	Поле «Пароль»	КЕ	Очікуваний результат
Т1-01	admin	admin	КЕ1, КЕ4	Головне вікно відкривається. У заголовку відображається роль [DIRECTOR].

Тести для неправильних класів еквівалентності функції «Автентифікація» наведено у таблиці 2.6.

Таблиця 2.6 – Тести для неправильних класів еквівалентності функції «Автентифікація» (В-01)

ID тесту	Поле «Логін»	Поле «Пароль»	КЕ	Очікуваний результат
1	2	3	4	5
Т1-02	(пусто)	admin	КЕ2	Головне вікно не відкривається. Відображається повідомлення про помилку.
Т1-03	unknown_user	admin	КЕ3	Головне вікно не відкривається. Відображається повідомлення «Помилка входу».

Продовження таблиці 2.6

1	2	3	4	5
Т1-04	Admin	(пусто)	КЕ5	Головне вікно не відкривається. Відображається повідомлення про помилку.
Т1-05	admin	wrongpass	КЕ6	Головне вікно не відкривається. Відображається повідомлення «Невірний пароль».

Тест 2. Тестування варіанту використання «Додавання клієнта» (В-02).

Виділені класи еквівалентності для функції «Додавання клієнта» наведено у таблиці 2.7.

Таблиця 2.7 – Виділення класів еквівалентності функції «Додавання клієнта» (В-02)

№	Поле	Умова	Клас	Тип
7	Прізвище	Непорожній рядок довжиною 1-50 символів	КЕ7	ПКЕ
8	Прізвище	Порожній рядок	КЕ8	НКЕ
9	Прізвище	Рядок довжиною понад 50 символів	КЕ9	НКЕ
10	Ім'я	Непорожній рядок довжиною 1-50 символів	КЕ10	ПКЕ
11	Ім'я	Порожній рядок	КЕ11	НКЕ
12	Телефон	Рядок формату +?[0-9]{10,13}	КЕ12	ПКЕ
13	Телефон	Порожній рядок	КЕ13	НКЕ
14	Телефон	Рядок з буквами або < 10 цифр	КЕ14	НКЕ
15	Телефон	Рядок з кількістю цифр > 13	КЕ15	НКЕ

Тести для правильних класів еквівалентності

Тести для правильних класів еквівалентності функції «Додавання клієнта» наведено у таблиці 2.8.

Таблиця 2.8 – Тести для правильних класів еквівалентності функції «Додавання клієнта» (В-02)

ID тесту	Прізвище	Ім'я	По-батькові	Телефон	КЕ	Очікуваний результат
1	2	3	4	5	6	7
Т2-01	Іваненко	Олег	Петрович	+380501234567	КЕ7, КЕ10, КЕ12	Запис додано до таблиці. Відображається повідомлення про успіх.

Продовження таблиці 2.8

1	2	3	4	5	6	7
Т2-02	А	Б	–	0501234567	КЕ7, КЕ10, КЕ12	Запис додано (по-батькові необов'язкове).

Тести для неправильних класів еквівалентності

Тести для неправильних класів еквівалентності функції «Додавання клієнта» наведено у таблиці 2.9.

Таблиця 2.9 – Тести для неправильних класів еквівалентності функції «Додавання клієнта» (В-02)

ID тесту	Прізвище	Ім'я	Телефон	КЕ	Очікуваний результат
1	2	3	4	5	6
Т2-03	–	Олег	+380501234567	КЕ8	Запис не збережено. Повідомлення: «Прізвище є обов'язковим».
Т2-04	А...А (51 символ)	Олег	+380501234567	КЕ9	Запис не збережено. Повідомлення про перевищення довжини.
Т2-05	Іваненко	–	+380501234567	КЕ11	Запис не збережено. Повідомлення: «Ім'я є обов'язковим».
Т2-06	Іваненко	Олег	–	КЕ13	Запис не збережено. Повідомлення: «Телефон є обов'язковим».
Т2-07	Іваненко	Олег	abc	КЕ14	Запис не збережено. Повідомлення: «Некоректний формат номеру телефону».
Т2-08	Іваненко	Олег	38050123456789 (14 цифр)	КЕ15	Запис не збережено. Повідомлення про некоректний формат.

У зв’язку зі значним обсягом функціональних можливостей та кількістю модулів програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» у роботі наведено лише частину тестових сценаріїв. Водночас програмний застосунок було протестовано в повному обсязі. За результатами проведеного тестування

встановлено, що програмне забезпечення функціонує коректно та відповідає поставленим вимогам, зокрема валідації, обмеженню типів та форматів вхідних даних.

Випробовування програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»

Випробовування функції «Авторизація».

Запустивши застосунок, користувач побачить вікно авторизації. Для авторизації потрібно ввести коректні існуючі дані (рис 2.15).

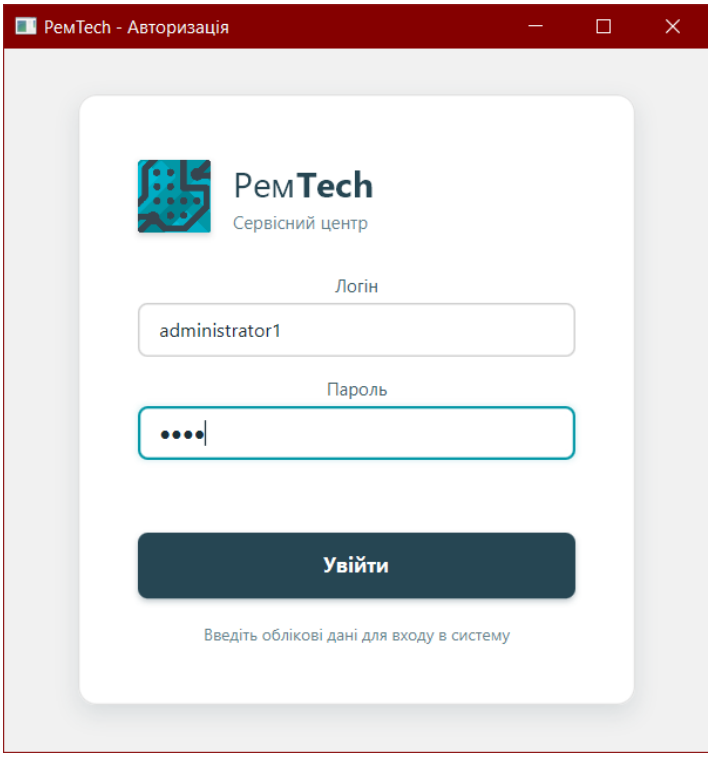


Рисунок 2.15 – Вікно «Авторизація»

Після, авторизації вікно авторизації буде змінено на вікно з головним меню застосунка, де наявні усі таблиці та функціонал згідно ролі облікового запису у який зайшов користувач.

Випробовування функції «Додавання запису про клієнта».

На вкладці «Клієнти» користувач вносить дані у форму (рис. 2.16).

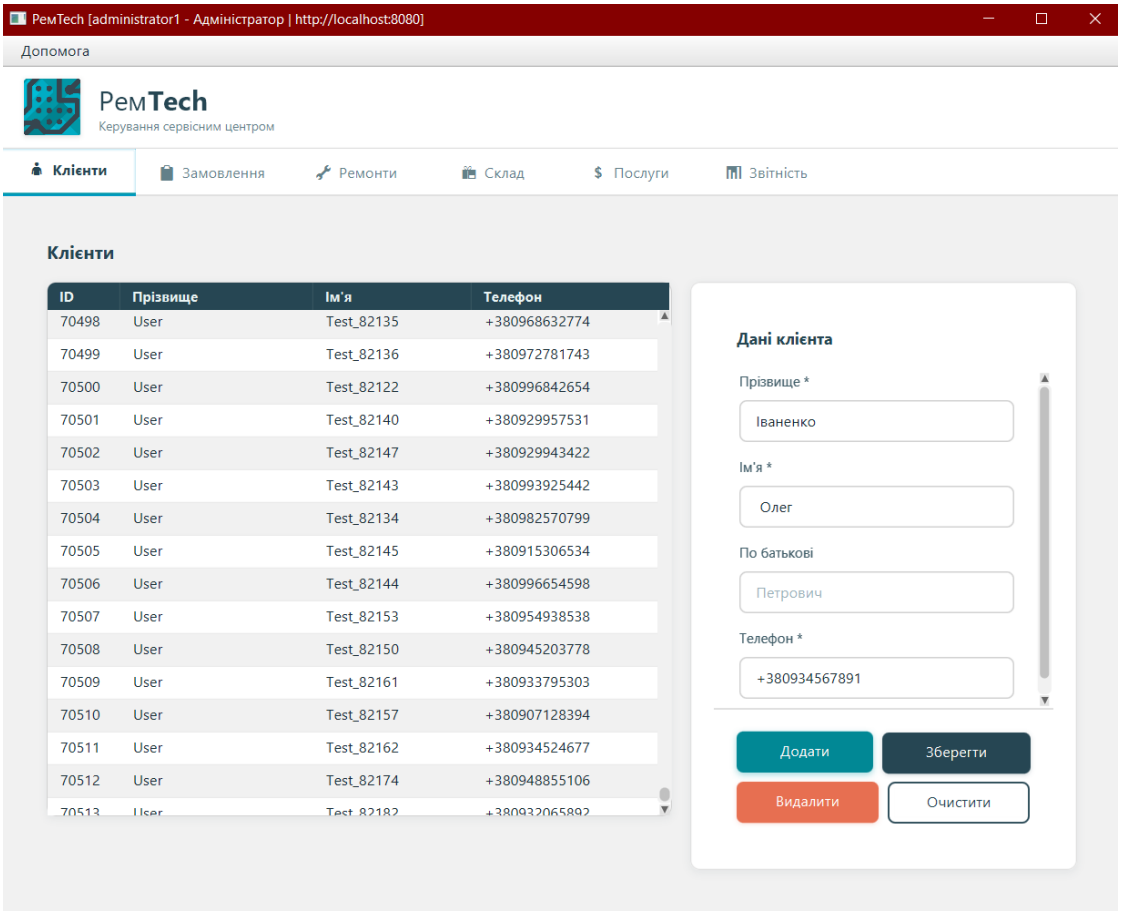


Рисунок 2.16 – Головне вікно застосунка, вкладка «Клієнти»

Користувач натискає «Додати» для підтвердження дії внесення даних. Після перевірки введених даних застосунком запис про нового клієнта буде створено (рис. 2.17).

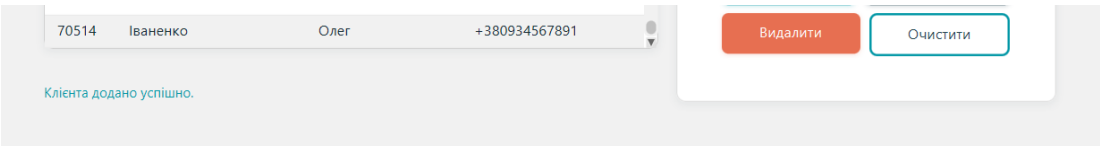


Рисунок 2.17 – Результат додавання інформації про клієнта

Робота інших функцій здійснюється за аналогічним принципом. За таким самим алгоритмом реалізовано функції додавання даних про замовлення,

ремонти, профілі, складські запаси та послуги. Рисунки відповідного функціоналу наведені в Додатку Г.

Для роботи зі звітністю треба перейти на вкладку «Звітність», обрати тип звіту, заповнити відповідну форму, та натиснути «Формувати звіт». Звіт буде сформовано справа від панелі. Після формування є можливість збереження звіту у форматах: CSV, XSLX, PDF (рис. 2.18).

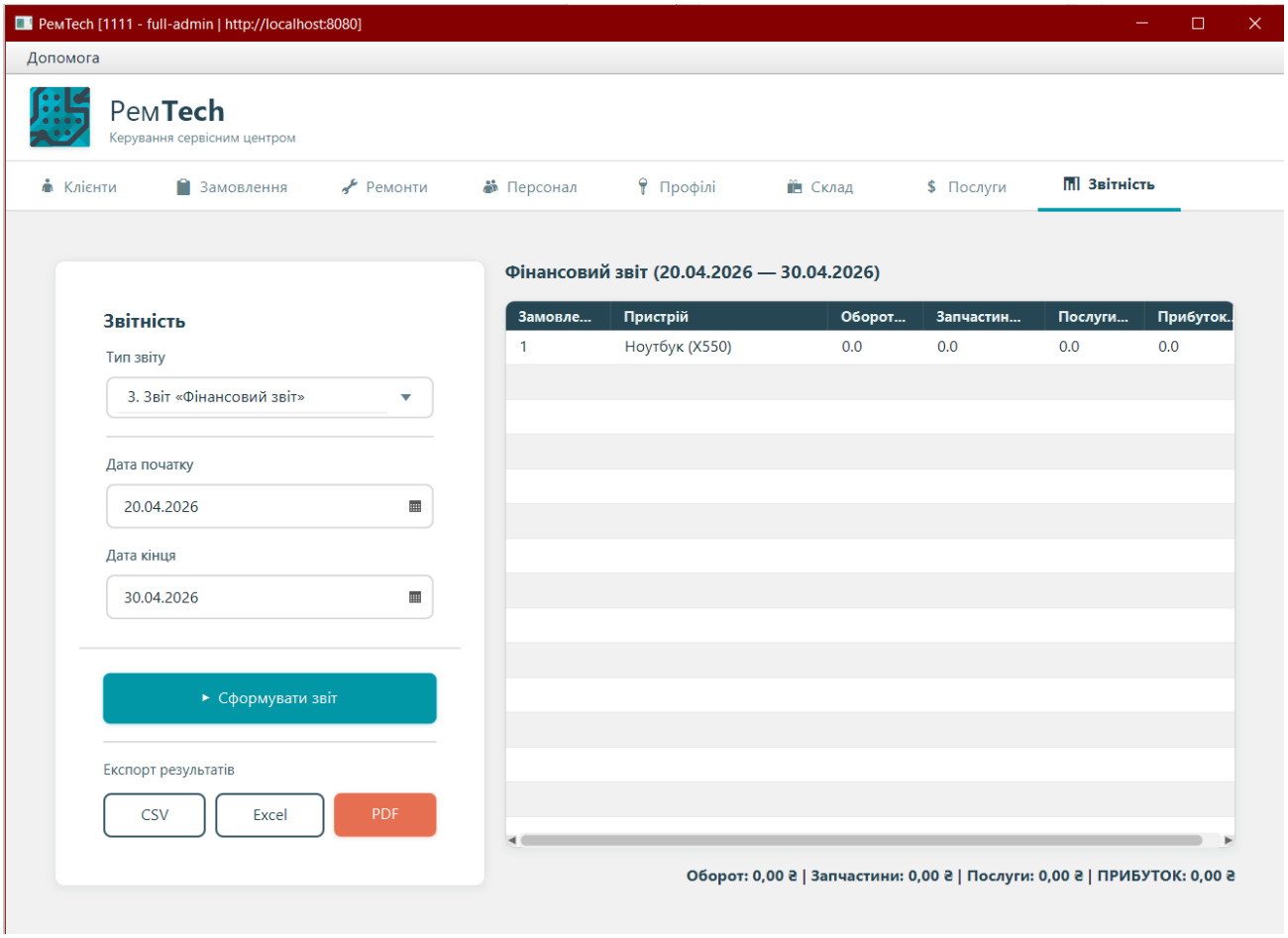


Рисунок 2.18 – Результат формування звіту

Під час проведення всіх випробувань, наявність помилок не виявлено, поведінка виконуваних функцій відповідає очікуваному результату. Розроблене програмне забезпечення для автоматизації обліку виконання робіт сервісного центру «РемTech» є працездатним та відповідає поставленим вимогам.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

У результаті виконання кваліфікаційної роботи розроблено програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech». Застосунок реалізовано у вигляді двох незалежних компонентів – серверної та клієнтської частин, які взаємодіють між собою за допомогою REST API.

Розроблене програмне забезпечення відповідає вимогам, визначеним у технічному завданні (Додаток А). Реалізовано всі дев'ять груп функцій:

- Автентифікація – вхід за логіном і паролем; роль визначається на стороні сервера та передається клієнту при успішному вході.
- Управління клієнтами – повний CRUD: додавання, редагування, видалення та перегляд записів про клієнтів.
- Управління замовленнями – повний CRUD: реєстрація пристрою з прив'язкою до клієнта та приймальника, контроль дат прийому та видачі.
- Управління ремонтами – повний CRUD: призначення майстрів, статусів, запчастин і послуг; автоматичний розрахунок вартості.
- Управління запчастинами – повний CRUD: облік найменувань, артикулів, кількості на складі та цінових параметрів.
- Управління послугами – повний CRUD: ведення прайс-листа послуг.
- Управління співробітниками – повний CRUD: облік кадрових даних персоналу.
- Формування звітності – чотири типи звітів (Акт приймання, Акт видачі, Фінансовий звіт, Використані запчастини) з можливістю експорту у форматах PDF, Excel (.xlsx) та CSV.
- Управління обліковими записами – повний CRUD: створення профілів із призначенням ролі (Майстер, Адміністратор, Директор).

Розмежування доступу реалізовано на рівні клієнтського застосунку: кнопки редагування, додавання та видалення приховуються залежно від ролі активного профілю. Вкладки, недоступні для поточної ролі, не відображаються у головному вікні.

Серверна частина побудована із використанням фреймворку Spring Boot 3 з використанням H2 як вбудованої реляційної СУБД у файловому режимі. База даних містить сім таблиць сутностей, між якими встановлено необхідні зв'язки: «один-до-багатьох» (клієнт – замовлення, замовлення – ремонт) та «багато-до-багатьох» (ремонт – майстри, ремонт – запчастини, ремонт – послуги). Вхідні дані валідуються на рівні DTO за допомогою анотацій Jakarta Validation. При першому запуску сервер автоматично створює обліковий запис з роллю «Директор».

Клієнтська частина реалізована із використанням JavaFX з архітектурним патерном MVVM. Застосунок містить вісім функціональних вкладок. Кожна таблиця оновлюється автоматично, що забезпечує актуальність даних при одночасній роботі кількох клієнтів. Адреса сервера зчитується з файлу config.properties.

Тестування програмного забезпечення виконано методом класів еквівалентності (Equivalence Class Partitioning). Для функцій виділено правильні та неправильні класи еквівалентності за всіма вхідними полями. Виявлених критичних дефектів у ході тестування не зафіксовано. Програмне забезпечення функціонує відповідно до визначених вимог.

Розроблена необхідна програмна документація: Технічне завдання – Додаток А, Код програми – Додаток Б, Опис програми – Додаток В, Інструкція користувача – Додаток Г, Програма та методика випробувань – Додаток Д.

Код розробленого програмного забезпечення «РемTech» складається з 4941 рядків коду мови Java. Застосунок серверної частини займає 55,5 МБ, клієнтської – 22,09 МБ.

4 ОХОРОНА ПРАЦІ

Розробка та експлуатація програмного забезпечення «РемTech» здійснюється в умовах офісного приміщення (сервісного центру) з використанням комп’ютерної техніки. У таких умовах праці основне навантаження на працівника пов’язане з тривалою роботою за комп’ютером, що зумовлює необхідність забезпечення безпечного та комфортного робочого середовища.

Метою цього розділу є проведення комплексного аналізу умов праці при роботі з комп’ютерною технікою, виявлення потенційно небезпечних і шкідливих виробничих факторів, а також розробка заходів щодо їх мінімізації або усунення відповідно до чинного законодавства України, зокрема Закону України «Про охорону праці» [23].

4.1 Характеристика робочого місця та аналіз шкідливих виробничих факторів при роботі з комп’ютерною технікою

Робоче місце розробника або користувача програмного застосунку «РемTech» розташоване в приміщенні сервісного центру площею 20 м² та об’ємом 60 м³, що відповідає загальним вимогам до організації робочих місць з використанням комп’ютерної техніки. Робоче місце оснащене столом, ергономічним кріслом, персональним комп’ютером з рідкокристалічним монітором, а також необхідними периферійними пристроями.

Організація робочого місця має суттєвий вплив на продуктивність праці, рівень втомлюваності працівника та ризик виникнення професійних захворювань. Тому важливо враховувати не лише технічне оснащення, а й умови

навколишнього середовища.

Згідно з НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями» [24], при роботі з комп’ютерною технікою на працівника можуть впливати такі групи шкідливих і небезпечних факторів.

Фізичні фактори:

- підвищений рівень електромагнітного та статичного випромінювання;
- небезпека ураження електричним струмом при роботі з обладнанням під напругою 220 В;
- невідповідність параметрів мікроклімату встановленим нормам;
- недостатній рівень освітленості робочої зони або наявність відблисків на екрані монітора;
- підвищений рівень шуму, що створюється вентиляторами системних блоків та кліматичним обладнанням.

Психофізіологічні фактори:

- підвищене навантаження на зоровий апарат унаслідок тривалої роботи з екраном;
- нервово-емоційне напруження, пов’язане з обробкою інформації та прийняттям рішень;
- статичне навантаження на опорно-руховий апарат (м’язи шиї, спини та рук) через тривале перебування в сидячому положенні.

Таким чином, навіть у відносно безпечних офісних умовах існує комплекс факторів, що можуть негативно впливати на стан здоров’я працівника, що обґрунтовує необхідність впровадження відповідних профілактичних заходів.

4.2 Заходи щодо нормалізації умов праці при роботі з комп’ютерною технікою

Для забезпечення безпечних і комфортних умов праці необхідно впровадити комплекс організаційних та технічних заходів, спрямованих на мінімізацію впливу виявлених шкідливих факторів.

Для забезпечення комфортних умов у приміщенні встановлено систему кондиціонування та припливно-витяжну вентиляцію. Згідно з ДСН 3.3.6.042-99 [25], для категорії робіт 1а (легка фізична робота), підтримуються такі оптимальні показники мікроклімату:

- температура повітря: у теплий період року +23-25 °С, у холодний +22-24 °С;
- відносна вологість повітря: 40–60%;
- швидкість руху повітря: не більше 0,1 м/с.

Дотримання цих показників сприяє зниженню втомлюваності працівника та підтриманню його працездатності протягом робочого дня.

Важливим фактором є також раціональна організація освітлення. У приміщенні застосовується комбінована система освітлення (природне та штучне), що дозволяє забезпечити достатній рівень освітленості протягом усього робочого часу.

Згідно з ДБН В.2.5-28:2018 [26], рівень освітленості робочої поверхні має становити не менше 400 лк. Для цього використовуються сучасні світлодіодні світильники, які забезпечують рівномірний світловий потік без мерехтіння та мінімізують появу відблисків на екранах моніторів. Робочі місця організовані таким чином, щоб природне світло падало переважно з лівого боку.

З метою зниження рівня шуму, який згідно з ДСН 3.3.6.037-99 не повинен перевищувати 50 дБА [27], використовуються малошумні компоненти комп’ютерної техніки та звукопоглинальні матеріали в оздобленні приміщення.

Окрему увагу приділено ергономіці робочого місця. Висота столу становить 720 мм, крісло є регульованим, що дозволяє адаптувати його під індивідуальні особливості працівника. Відстань від очей до монітора становить 60–70 см, що відповідає нормативним вимогам.

Для зменшення зорового та нервово-емоційного навантаження запроваджено регламентовані перерви: 15 хвилин після кожних 2 годин роботи за комп’ютером. Це сприяє профілактиці перевтоми та зниженню ризику професійних захворювань.

4.3 Електробезпека при роботі з комп’ютерною технікою

Приміщення сервісного центру відноситься до приміщень без підвищеної небезпеки ураження електричним струмом, оскільки воно є сухим, має нормальний температурний режим та неструмопровідну підлогу. Однак навіть за таких умов наявність значної кількості електронно-обчислювальної техніки, периферійних пристроїв та допоміжного обладнання зумовлює необхідність суворого дотримання вимог електробезпеки відповідно до Правил улаштування електроустановок (ПУЕ) [28].

Електробезпека є одним із ключових аспектів організації безпечних умов праці, оскільки ураження електричним струмом може призвести до серйозних травм, порушення роботи обладнання або виникнення пожежонебезпечних ситуацій. Основними причинами виникнення небезпечних ситуацій є пошкодження ізоляції кабелів, неправильне підключення обладнання, перевантаження електромережі та недотримання правил експлуатації.

Для забезпечення належного рівня електробезпеки в приміщенні сервісного центру передбачено комплекс технічних і організаційних заходів.

До основних технічних заходів належать:

- використання трипровідної системи електроживлення (фаза, нуль, захисний провідник), що забезпечує ефективне заземлення обладнання;
- обов’язкове занулення та заземлення металевих корпусів системних блоків, серверного обладнання та периферійних пристроїв, що дозволяє уникнути накопичення небезпечного потенціалу на їх поверхні;
- встановлення пристроїв захисного відключення (ПЗВ), які автоматично відключають живлення у разі виникнення струму витоку понад 30 мА, що суттєво знижує ризик ураження людини;
- використання автоматичних вимикачів для захисту електромережі від перевантажень і коротких замикань.

Важливу роль відіграє також стан електропроводки та кабельної інфраструктури. Усі струмопровідні частини мають бути надійно ізольовані, а кабелі – прокладені таким чином, щоб виключити можливість їх механічного пошкодження. Регулярно проводиться візуальний огляд проводів, розеток і з’єднань з метою виявлення дефектів ізоляції або ознак перегріву.

Крім технічних заходів, необхідно дотримуватися організаційних вимог. Працівники повинні бути ознайомлені з правилами безпечної експлуатації електрообладнання, проходити відповідні інструктажі та не допускатися до роботи з несправною технікою. Забороняється самостійне втручання в електричні схеми обладнання без відповідної кваліфікації.

Додатково рекомендується уникати перевантаження електромережі шляхом раціонального розподілу навантаження між розетками та використання сертифікованих подовжувачів і мережевих фільтрів із захистом від перенапруги. Це особливо актуально для сервісних центрів, де одночасно може працювати велика кількість пристроїв.

Таким чином, комплексне дотримання вимог електробезпеки дозволяє мінімізувати ризик ураження електричним струмом, забезпечити надійну роботу обладнання та створити безпечні умови праці для персоналу.

4.4 Пожежна безпека при роботі з комп’ютерною технікою

Пожежна безпека є невід’ємною складовою організації безпечних умов праці в сервісному центрі, оскільки використання електрообладнання та наявність горючих матеріалів створюють потенційну загрозу виникнення пожежі.

Приміщення, у якому експлуатується програмне забезпечення «РемTech», відноситься до категорії пожежної небезпеки «В», що обумовлено наявністю твердих горючих матеріалів, таких як пластик корпусів обладнання, ізоляція кабелів, паперова документація та пакувальні матеріали.

Відповідно до вимог НАПБ А.01.001-2014 [29], для забезпечення належного рівня пожежної безпеки в приміщенні реалізується комплекс організаційних і технічних заходів.

До організаційних заходів належать:

- проведення вступних, первинних та періодичних інструктажів з пожежної безпеки для працівників;
- призначення відповідальної особи за пожежну безпеку, яка контролює дотримання встановлених вимог;
- розробка та розміщення на видимих місцях планів евакуації у разі виникнення пожежі;
- встановлення чітких правил поводження з електрообладнанням і заборона використання несправних пристроїв.

Технічні заходи спрямовані на своєчасне виявлення пожежі та її ліквідацію на початковій стадії. До них належать:

- оснащення приміщення автоматичною пожежною сигналізацією з димовими сповіщувачами, що забезпечують раннє виявлення загоряння;
- використання автоматичних вимикачів для захисту електромережі від перевантажень і коротких замикань, які є однією з основних причин пожеж;

- забезпечення робочих місць первинними засобами пожежогасіння.

Особливу увагу приділено вибору типу вогнегасників. Оскільки у приміщенні експлуатується електронне обладнання, застосування водяних або пінних засобів пожежогасіння є неприпустимим через ризик ураження електричним струмом та пошкодження техніки. Тому використовуються вуглекислотні вогнегасники (типу ВВК-2 або ВВК-3), які не проводять електричний струм і не залишають слідів на поверхнях електронних компонентів.

Крім того, важливим є дотримання правил експлуатації приміщення. Забороняється захаращення евакуаційних шляхів, зберігання легкозаймистих матеріалів поблизу джерел тепла та використання пошкодженого електрообладнання. Всі працівники повинні знати порядок дій у разі виникнення пожежі, включаючи правила користування вогнегасниками та маршрути евакуації.

Профілактичні заходи також включають регулярну перевірку стану електромережі, пожежної сигналізації та засобів пожежогасіння. Це дозволяє своєчасно виявляти потенційні несправності та запобігати виникненню надзвичайних ситуацій.

Отже, дотримання встановлених норм і правил пожежної безпеки, а також впровадження відповідних технічних і організаційних заходів забезпечує зниження ризику виникнення пожежі та гарантує безпеку персоналу і матеріальних цінностей сервісного центру.

Висновки до розділу

У результаті проведеного аналізу умов праці на робочому місці користувача програмного забезпечення «РемТech» визначено основні шкідливі та небезпечні фактори, характерні для роботи з комп’ютером.

Запропоновані заходи щодо нормалізації мікроклімату, забезпечення

належного рівня освітлення (не менше 400 лк), дотримання ергономічних вимог, а також впровадження засобів електро- та пожежної безпеки відповідають чинним нормативним документам.

Реалізація зазначених заходів забезпечує створення безпечних умов праці, сприяє збереженню здоров’я працівників та підтриманню їх високої працездатності під час роботи з програмним забезпеченням.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було виконано практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов - розробку програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech».

Для досягнення поставленої мети було виконано наступні завдання:

- проведено аналіз практичної задачі інженерії програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»;
- проведено огляд та аналіз існуючих програмних продуктів для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки;
- сформульовано постановку задачі та технічні вимоги на розробку програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»;
- проведено аналіз вимог до програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»;
- розроблено архітектуру програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech»;
- розроблено проєкт програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту техніки «РемTech»;
- виконано тестування та випробування програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної

техніки «РемТech»;

- розроблено програмну документацію для програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту техніки «РемТech».

Таким чином, поставлені завдання, які передбачала кваліфікаційна робота були виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gartner Forecasts Worldwide IT Spending to Grow 7.9% in 2025. URL: https://www.gartner.com/en/newsroom/press-releases/2025-07-15-gartner-forecasts-worldwide-it-spending-to-grow-7-point-9-percent-in-2025 (дата звернення: 04.02.2026).

2. Personal Computing Devices Market Insights. URL: https://www.idc.com/promo/pcdforecast/ (дата звернення: 04.02.2026).

3. Український IT-експорт зріс уперше за три роки. URL: https://delo.ua/news/ukrayinskii-it-eksport-zris-uperse-za-tri-roki-460484/ (дата звернення: 04.02.2026).

4. REST API: принципи роботи веб-сервісів. URL: https://wezom.com.ua/ua/blog/rest-api-printsipi-roboti-veb-servisiv (дата звернення: 05.02.2026).

5. RO App – Зручна програма для автоматизації бізнесу. URL: https://roapp.io/uk/ (дата звернення 11.02.2026).

6. Універсальна Система Обліку – Облік сервісного центру. URL: https://ususoft.com.ua/uk/uchet_servisnogo_tsentra.php (дата звернення: 09.02.2026).

7. Gincore – Управління замовленнями, продажами, клієнтами, фінансами вашої компанії. URL: https://gincore.net/uk (дата звернення: 11.02.2026).

8. What is CRUD? Explained. URL: https://www.codecademy.com/article/what-is-crud-explained (дата звернення: 19.02.2026).

9. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston : Addison-Wesley, 2005. 496 p.

10. Introduction to Model View View Model (MVVM). URL: https://www.geeksforgeeks.org/websites-apps/introduction-to-model-view-view-model-mvvm/ (дата звернення: 25.02.2026).

11. Understanding the Layered Architecture Pattern: A Comprehensive Guide. URL: https://dev.to/yasmine_ddec94f4d4/understanding-the-layered-architecture-pattern-a-comprehensive-guide-1e2j (дата звернення: 28.02.2026).

12. Створення логічної моделі даних. URL: https://ddm-architecture-cluster-mgmt.apps.krrt-stage.ncr.gov.ua/ua/platform/1.9.7/registry-develop/data-modeling/data/logical-model/data-modelling-logical-datamodel.html (дата звернення: 03.03.2026).

13. Java In Use Today & In the Future. URL: https://education.oracle.com/en/java-in-use-2020-beyond (дата звернення: 06.03.2026).

14. Schildt H. Java: The Complete Reference. 9th ed. New York : McGraw-Hill Education, 2014. URL: https://www.sietk.org/downloads/javabook.pdf (дата звернення: 09.03.2026).

15. ACID Properties in DBMS. URL: https://www.geeksforgeeks.org/dbms/acid-properties-in-dbms/ (дата звернення: 12.03.2026).

16. Building REST services with Spring. URL: https://spring.io/guides/tutorials/rest (дата звернення: 15.03.2026).

17. Spring Data JPA. URL: https://docs.spring.io/spring-data/jpa/reference/ (дата звернення: 18.03.2026).

18. Jakarta Bean Validation specification. URL: https://jakarta.ee/specifications/bean-validation/3.0/jakarta-bean-validation-spec-3.0.html (дата звернення: 21.03.2026).

19. Oracle JavaFX. URL: https://www.oracle.com/javase/javafx/ (дата звернення: 24.03.2026).

20. Концептуальні, логічні та фізичні моделі даних . URL: https://data-life-ua.com/db/kontseptualni-lohichni-ta-fizychni-modeli-danykh/ (дата звернення: 27.03.2026).

21. Чому SOLID – важлива складова мислення програміста. URL: https://dou.ua/lenta/articles/solid-principles/ (дата звернення: 30.03.2026).

22. Класи еквівалентності, граничні значення. URL: https://training.qatestlab.com/blog/technical-articles/equivalence-classes-and-boundary-values/ (дата звернення: 02.04.2026).

23. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. Редакція від 12.09.2025. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 20.02.2026).

24. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Наказ Міністерства соціальної політики України від 14.02.2018 № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 20.02.2026).

25. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень : Затверджено постановою Головного державного санітарного лікаря України від 01.12.1999 № 42. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 21.02.2026).

26. ДБН В.2.5-28:2018. Природне і штучне освітлення : Наказ Міністерства регіонального розвитку, будівництва та житлово- комунального господарства України 03.10.2018 № 264 URL: https://e-construction.gov.ua/laws_detail/3074958732556240833 (дата звернення: 23.02.2026).

27. ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку : Затверджено постановою Головного державного санітарного лікаря України від 01.12.1999 № 37. URL: https://zakon.rada.gov.ua/rada/show/va037282-99 (дата звернення: 23.02.2026).

28. Про затвердження Правил улаштування електроустановок : Затверджено наказом Міненерговугілля України від 21.07.2017 № 476. URL: https://zakon.rada.gov.ua/rada/show/v0476732-17 (дата звернення: 26.02.2026).

29. Про затвердження Правил пожежної безпеки в Україні : Затверджено наказом МВС України від 30.12.2014 № 1417. Редакція від 27.02.2026. URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 27.02.2026).

Додаток А – Технічне завдання на розробку програмного забезпечення «РемTech»

Програмним продуктом є програмний застосунок для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Сфера застосування

Програмний застосунок «РемTech» використовується у сервісному центрі «РемTech», для забезпечення ефективної роботи та задоволення потреб сервісного центру та клієнтів.

1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на кваліфікаційну роботу на здобуття ступеня вищої освіти "бакалавр".

2 Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням програмного застосунку «РемTech» є автоматизація обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech», яка покликана вирішити проблему комплексності та невизначеності обраної задачі інженерії програмного забезпечення, та надати інструмент автоматизації повного циклу обслуговування клієнтів – від первинної реєстрації несправного пристрою та клієнта до формування звітності.

2.2 Функціональне призначення

Для адміністратора:

- Управління клієнтами: адміністратор повинен мати можливість створювати, редагувати та видаляти записи про клієнтів.
- Управління замовленнями: адміністратор повинен мати можливість

створювати, редагувати та видаляти записи замовлень.

- Перегляд ремонтів: адміністратор повинен мати можливість лише переглядати інформацію ремонту.
- Перегляд запчастин: адміністратор повинен мати можливість лише переглядати інформацію про запчастини.
- Перегляд послуг: адміністратор повинен мати можливість лише переглядати інформацію послуги.
- Формування звітів: адміністратор може формувати звіт «Акт приймання пристрою» (8.1) та звіт «Акт видачі пристрою» (8.2).

Для майстра:

- Перегляд замовлень: майстер повинен мати можливість лише переглядати інформацію про замовлення.
- Управління запчастинами: майстер повинен мати можливість редагувати та видаляти інформацію щодо запчастин або їх деталей, додавати нові або змінювати існуючі.
- Перегляд послуг: майстер повинен мати можливість лише переглядати інформацію про послуги.
- Перегляд клієнтів: майстер повинен мати можливість лише переглядати записи про клієнтів.
- Управління ремонтами: майстер може створювати, редагувати та видаляти записи про ремонти, інформацію про хід ремонту, додавати використані запчастини та змінювати статус замовлення.
- Формування звітів: майстер може формувати звіт «Використані запчастини» (8.4)

Для директора:

- Управління співробітниками: директор повинен мати можливість створювати, редагувати та видаляти записи про працівників, їх контактні дані, посади.

- Перегляд запчастин: директор повинен мати можливість лише переглядати інформацію про запчастини.
- Управління послугами: директор або власник повинні мати можливість створювати, редагувати та видаляти записи про послуги.
- Перегляд замовлень: директор повинен мати можливість лише переглядати інформацію про замовлення.
- Перегляд клієнтів: директор повинен мати можливість лише переглядати записи про клієнтів.
- Формування звітів: директор може формувати звіт «Фінансовий звіт» (8.3) та «Використані запчастини» (8.4).
- Управління обліковими записами: директор повинен мати можливість створювати, редагувати та видаляти облікові записи для входу в систему з призначенням відповідної ролі (Адміністратор, Майстер, Директор).

3 Вимоги до програмного забезпечення

Програмне забезпечення повинно містити:

- Серверну частину застосунку.
- Клієнтську частину застосунку.

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинне виконувати наступні функції:

1) Автентифікація

Вхідні: логін, пароль.

Вихідні: повідомлення про автентифікацію.

2) Дані клієнта

2.1) Додавання клієнта

Вхідні: прізвище, ім’я, по-батькові, контактні дані.

Вихідні: повідомлення про результат додавання даних про клієнта.

2.2) Редагування клієнта

Вхідні: ідентифікатор клієнта, прізвище, ім’я, по-батькові, контактні дані.

Вихідні: повідомлення про результат оновлення даних клієнта.

2.3) Видалення клієнта

Вхідні: ідентифікатор клієнта.

Вихідні: повідомлення про результат видалення даних про клієнта.

2.4) Перегляд клієнтів

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список клієнтів.

3) Дані замовлення

3.1) Створення замовлення

Вхідні: назва пристрою, виробник, модель, serial number (SNID), опис стану, опис проблеми, дата прийому, орієнтована дата видачі, дата видачі, приймальник, власник.

Вихідні: повідомлення про результат створення замовлення.

3.2) Редагування замовлення

Вхідні: ідентифікатор замовлення, назва пристрою, виробник, модель, serial number (SNID), опис стану, опис проблеми, дата прийому, орієнтована дата видачі, дата видачі, приймальник, власник.

Вихідні: повідомлення про результат оновлення даних замовлення.

3.3) Видалення замовлення

Вхідні: ідентифікатор замовлення.

Вихідні: повідомлення про результат видалення замовлення.

3.4) Перегляд замовлень

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список замовлень.

4) Дані ремонту

4.1) Додавання ремонту

Вхідні: номер замовлення, дата початку ремонту, дата кінця ремонту, майстер, послуги, запчастини, результат діагностики, вартість ремонту, статус ремонту.

Вихідні: повідомлення про результат додавання даних про ремонт.

4.2) Редагування даних про ремонт

Вхідні: ідентифікатор ремонту, номер замовлення, дата початку ремонту, дата кінця ремонту, майстер, послуги, запчастини, результат діагностики, вартість ремонту, статус ремонту.

Вихідні: повідомлення про результат оновлення даних ремонту.

4.3) Видалення даних про ремонт

Вхідні: ідентифікатор ремонту.

Вихідні: повідомлення про результат видалення даних про ремонт.

4.4) Перегляд ремонтів

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список ремонтів.

5) Дані співробітників

5.1) Додавання співробітника

Вхідні: посада, прізвище, ім’я, по-батькові, контактний номер, кваліфікація, адреса, дата початку роботи, дата звільнення.

Вихідні: повідомлення про результат додавання даних про робітника.

5.2) Редагування співробітника

Вхідні: ідентифікатор робітника, посада, прізвище, ім’я, по-батькові, контактний номер, кваліфікація, адреса, дата початку роботи, дата звільнення.

Вихідні: повідомлення про результат оновлення даних робітника.

5.3) Видалення співробітника

Вхідні: ідентифікатор робітника, дата звільнення.

Вихідні: повідомлення про результат видалення робітника.

5.4) Перегляд робітників

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список робітників.

6) Дані послуг

6.1) Додавання послуги

Вхідні: назва послуги, опис послуги, вартість послуги.

Вихідні: повідомлення про результат додавання даних про послуги.

6.2) Редагування послуги

Вхідні: ідентифікатор послуги, назва послуги, опис послуги, вартість послуги.

Вихідні: повідомлення про результат оновлення даних послуги.

6.3) Видалення послуги

Вхідні: ідентифікатор послуги.

Вихідні: повідомлення про результат видалення послуги.

6.4) Перегляд послуг

Вхідні: параметри пошуку та фільтрації.

Вихідні: сформований список послуг.

7) Дані запчастин

7.1) Додавання запчастини

Вхідні: назва запчастини, артикул запчастини, вартість покупки, вартість для ремонту, кількість.

Вихідні: повідомлення про результат додавання даних про запчастину

7.2) Редагування запчастини

Вхідні: ідентифікатор запчастини, артикул запчастини, назва запчастини, вартість покупки, вартість для ремонту, кількість.

Вихідні: повідомлення про результат оновлення даних запчастини.

7.3) Видалення запчастини

Вхідні: ідентифікатор запчастини.

Вихідні: повідомлення про результат видалення даних запчастини.

7.4) Перегляд запчастини

Вхідні: параметри пошуку та фільтрації

Вихідні: сформований список запчастин.

8) Звітність

8.1) Звіт «Акт приймання»

Вхідні: тип звіту, номер замовлення

Вихідні: Звіт «Акт приймання» з інформацією про конкретне замовлення, клієнта та деталями про його пристрій.

8.2) Звіт «Акт видачі»

Вхідні: тип звіту, номер замовлення

Вихідні: звіт «Акт видачі» з інформацією та деталями про конкретне замовлення, клієнта та ремонту, що до нього належать.

8.3) Звіт «Фінансовий звіт»

Вхідні дані: тип звіту, початкова дата, кінцева дата.

Вихідні дані: звіт «Фінансовий звіт» з інформацією: список замовлень з виконаним статусом, з їх фінансовими даними за введений проміжок часу (оборот, витрати на запчастини, вартість послуг, прибуток).

8.4) Звіт «Використані запчастини»

Вхідні дані: тип звіту, початкова дата, кінцева дата.

Вихідні дані: звіт «Використані запчастини» з інформацією: список ремонтів за введений проміжок часу з інформацією про використання запчастин (витрачені запчастини, кількість, залишок, ціна закупки, ціна продажу, маржа).

9) Дані облікового запису

9.1) Додавання облікового запису

Вхідні: логін, пароль, роль.

Вихідні: повідомлення про результат додавання даних про обліковий запис.

9.2) Редагування облікового запису

Вхідні: ідентифікатор облікового запису, логін, пароль, роль.

Вихідні: повідомлення про результат оновлення даних облікового запису.

9.3) Видалення облікового запису

Вхідні: номер облікового запису.

Вихідні: повідомлення про результат видалення даних облікового запису.

9.4) Перегляд облікового запису

Вхідні: параметри пошуку та фільтрації

Вихідні: сформований список облікових записів.

3.2 Вимоги до організації вхідних та вихідних даних

Вхідною інформацією є дані, які вводяться вручну у поля форм, або обираються з представлених списків. Залежно від типу поля, дані можуть бути: числові, номери, текстові, дати, або варіанти представлені списком.

Вихідними даними є результати виконаних операцій, які відображаються у вигляді екранних форм, повідомлень, записів або звітів.

3.3 Вимоги до надійності

3.3.1 Вимоги до забезпечення надійного (сталого) функціонування програмного забезпечення

Резервне копіювання (Backup): програмне забезпечення повинно забезпечувати регулярне резервне копіювання бази даних, щоб уникнути втрати інформації через збої в системі чи апаратному забезпеченні.

Збереження стану: програмне забезпечення повинно мати функціонал автоматичного відновлення роботи після аварійного завершення. Після несподіваного закриття програми повинна бути можливість продовжити роботу з того місця, де сталася помилка.

Контроль доступу: надійність функціонування також забезпечується захистом від несанкціонованого втручання в структуру бази даних через систему авторизації та розмежування прав.

3.3.2 Відмови через некоректні дії оператора

Валідація даних: програмне забезпечення повинно перевіряти всі вхідні дані на коректність та заповнення обов’язкових полів перед їх обробкою. Перевірка форматів для полів: з цифрами – заблокувати введення букв; для номерів – обмежити за допомогою маски «+380 (000) 000-00-00»; для дати – обмежити введення лише існуючих значень у форматі «дд.мм.рррр»; для вибору зі списку – обмеження надані виключно варіантами представленим даним списком.

Блокування дій: функціональні кнопки керування даними мають залишатися неактивними, або надавати сповіщення про некоректні дані або пусті обов’язкові поля, доки всі обов'язкові поля запису не пройдуть валідацію.

Сповіщення про помилки: якщо користувач вводить некоректні дані, програма повинна відразу сповістити про це користувача за допомогою повідомлення або візуальною індикацією. Повідомлення має бути чітким і зрозумілим, щоб користувач міг швидко виправити помилку вводу даних.

3.4 Умови експлуатації

3.4.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.4.2 Вимоги до чисельності та кваліфікації користувача

Три категорії користувачів мають працювати з програмним застосунком, користувачі повинні мати базові навички роботи з комп'ютером.

3.5 Вимоги до складу і параметрів технічних засобів

Система повинна відповідати таким мінімальним характеристикам комп’ютера:

- Процесор: двоядерний процесор Intel або AMD, з тактовою частотою 2 ГГц.
- Оперативна пам’ять: 4 ГБ.
- Дисковий простір: HDD/SSD на 128 ГБ.
- Мережеві можливості: мережевий адаптер з підтримкою Ethernet для дротової локальної мережі, або підтримка WIFI для бездротової локальної мережі.

3.6 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно мати повну сумісність з однією з операційних систем: Windows 10 (версії 64-bit) та вище, macOS 10.x та вище або Linux (x86-64/AArch64). Це включає коректну роботу всіх функцій програмного забезпечення, включаючи установку, запуск, оновлення та взаємодію з іншими системними компонентами ОС.

3.7 Спеціальні вимоги

Спеціальні вимоги не висуваються.

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програму та методику випробувань;
- керівництво користувача;
- опис програми;
- код програми.

5 Техніко-економічні показники

Техніко-економічні показники не розраховуються.

6 Етапи роботи

Етапи та терміни виконання робіт представлені у таблиці А.1.

Таблиця А.1 – Етапи та терміни виконання робіт

Номер	Назва етапів роботи	Термін виконання
1	Аналіз практичної задачі інженерії програмного забезпечення	12.02.2026
2	Аналіз вимог до програмного забезпечення	09.03.2026
3	Розробка архітектури програмного забезпечення	06.04.2026
4	Розробка проєкту програмного забезпечення	14.04.2026
5	Реалізація та тестування програмного забезпечення	30.04.2026

7 Порядок контролю та приймання

Контроль за аналізом та проєктуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у технічному завданні. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом проводиться відповідно до програми і методики випробувань і документується за допомогою протоколу проведення випробувань. У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – КОД ПРОГРАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

Лістинг 1 – ClientController.java

```
package max.server.controller;
import jakarta.validation.Valid;
import max.server.dto.ClientDto;
import max.server.service.ClientService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/clients")
public class ClientController {
    @Autowired
    private ClientService clientService;
    @PostMapping
    public ClientDto createClient(@Valid @RequestBody ClientDto
clientDto) {
        return clientService.createClient(clientDto);
    }
    @GetMapping
    public List<ClientDto> getAllClients() {
        return clientService.getAllClients();
    }
    @GetMapping("/{id}")
    public                              ResponseEntity<ClientDto>
getClientById(@PathVariable Long id) {
        try {
            return
ResponseEntity.ok(clientService.getClientById(id));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }
    @PutMapping("/{id}")
    public                              ResponseEntity<ClientDto>
updateClient(@PathVariable Long id,
                                                    @Valid
@RequestBody ClientDto clientDto) {
        try {
            return
ResponseEntity.ok(clientService.updateClient(id, clientDto));
        } catch (RuntimeException e) {
```

```
                return ResponseEntity.notFound().build();
            }
        }
        @DeleteMapping("/{id}")
        public ResponseEntity<Void> deleteClient(@PathVariable Long
id) {
            clientService.deleteClient(id);
            return ResponseEntity.noContent().build();
        }
}
```

Листинг 2 – ClientDto.java

```
package max.server.dto;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Pattern;
import jakarta.validation.constraints.Size;
import lombok.Data;
import java.io.Serializable;
@Data
public class ClientDto implements Serializable {
    private Long id;
    @NotBlank(message = "Прізвище обов'язкове")
    @Size(max = 50)
    private String lastName;
    @NotBlank(message = "Ім'я обов'язкове")
    @Size(max = 50)
    private String firstName;
    @Size(max = 50)
    private String middleName;
    private  String  fullName;  //  для  зручності  відображення
клієнта
    @NotBlank(message = "Номер телефону обов'язковий")
    @Pattern(regexp   =   "^\\+?[0-9]{10,13}$",   message   =
"Некоректний формат телефону")
    private String phoneNumber;
}
```

Листинг 3 –Client.java

```
package max.server.model;
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Pattern;
import jakarta.validation.constraints.Size;
import lombok.*;
import java.util.Set;
@Getter
@Setter
@NoArgsConstructor
@EqualsAndHashCode(of = "id")
@ToString(exclude = "orders")
```

```
@Entity
@Table(name = "clients")
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank(message = "Прізвище є обов'язковим")
    @Size(max = 50)
    @Column(nullable = false)
    private String lastName; // Прізвище
    @NotBlank(message = "Ім'я є обов'язковим")
    @Size(max = 50)
    @Column(nullable = false)
    private String firstName; // Ім'я
    @Size(max = 50)
    private String middleName; // По-батькові
    @NotBlank(message = "Номер телефону є обов'язковим")
    @Pattern(regexp = "^\\+?[0-9]{10,13}$", message =
"Некоректний формат номера телефону")
    @Column(name = "phone_number", unique = true, nullable =
false)
    private String phoneNumber; // Контактні дані
    // Один клієнт може мати багато замовлень
    @OneToMany(mappedBy = "client", cascade = CascadeType.ALL,
fetch = FetchType.LAZY)
    private Set<Order> orders;
    /**
     * Формує повне ім'я для відображення в інтерфейсі
(readonly)
     */
    public String getFullName() {
        return String.format("%s %s %s",
                lastName,
                firstName,
                (middleName != null ? middleName : "")).trim();
    }
}
```

Листинг 4 –ClientRepository.java

```
public interface ClientRepository extends JpaRepository<Client,
Long> { }
```

Листинг 5 –ClientService.java

```
package max.server.service;
import max.server.dto.ClientDto;
import max.server.model.Client;
import max.server.repository.ClientRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import
```

```
org.springframework.transaction.annotation.Transactional;
     import java.util.List;
     import java.util.stream.Collectors;
     @Service
     public class ClientService {
         @Autowired
         private ClientRepository clientRepository;
         @Transactional
         public ClientDto createClient(ClientDto dto) {
             Client client = dtoToEntity(dto);
             Client savedClient = clientRepository.save(client);
             return entityToDto(savedClient);
         }
         @Transactional(readOnly = true)
         public List<ClientDto> getAllClients() {
             return clientRepository.findAll()
                     .stream()
                     .map(this::entityToDto)
                     .collect(Collectors.toList());
         }

         @Transactional(readOnly = true)
         public ClientDto getClientById(Long id) {
             Client client = clientRepository.findById(id)
                     .orElseThrow(()              ->              new
RuntimeException("Клієнта з ID " + id + " не знайдено"));
             return entityToDto(client);
         }
         @Transactional
         public ClientDto updateClient(Long id, ClientDto dto) {
             Client client = clientRepository.findById(id)
                     .orElseThrow(()              ->              new
RuntimeException("Клієнта для оновлення не знайдено"));
             client.setFirstName(dto.getFirstName());
             client.setLastName(dto.getLastName());
             client.setMiddleName(dto.getMiddleName());
             client.setPhoneNumber(dto.getPhoneNumber());

             return entityToDto(clientRepository.save(client));
         }
         @Transactional
         public void deleteClient(Long id) {
             if (!clientRepository.existsById(id)) {
                 throw  new  RuntimeException("Неможливо  видалити:
клієнта не знайдено");
             }
             clientRepository.deleteById(id);
         }
         // Конвертери (Mapping)
```

```
        private ClientDto entityToDto(Client client) {
            ClientDto dto = new ClientDto();
            dto.setId(client.getId());
            dto.setFirstName(client.getFirstName());
            dto.setLastName(client.getLastName());
            dto.setMiddleName(client.getMiddleName());
            dto.setFullName(client.getFullName());                //
Використовуємо метод з моделі
            dto.setPhoneNumber(client.getPhoneNumber());
            return dto;
        }
        private Client dtoToEntity(ClientDto dto) {
            Client client = new Client();
            client.setFirstName(dto.getFirstName());
            client.setLastName(dto.getLastName());
            client.setMiddleName(dto.getMiddleName());
            client.setPhoneNumber(dto.getPhoneNumber());
            return client;
        }
}
```

Листинг 6 –ClientTabController.java

```
    package max.controller;
    import javafx.fxml.FXML;
    import javafx.scene.control.*;
    import javafx.scene.control.cell.PropertyValueFactory;
    import max.dto.ClientDto;
    import max.viewmodel.ClientViewModel;
    import max.viewmodel.MainViewModel.TabId;
    /**
     * Контролер для вкладки управління клієнтами.
     */
    public class ClientTabController extends BaseTabController {
        @FXML private TableView<ClientDto> clientTable;
        @FXML private TableColumn<ClientDto, Long> idColumn;
        @FXML     private     TableColumn<ClientDto,     String>
lastNameColumn;
        @FXML     private     TableColumn<ClientDto,     String>
firstNameColumn;
        @FXML private TableColumn<ClientDto, String> phoneColumn;
        @FXML private TextField lastNameField;
        @FXML private TextField firstNameField;
        @FXML private TextField middleNameField;
        @FXML private TextField phoneField;
        @FXML private Label errorLabel;
        @FXML private Label successLabel;
        @FXML private Button addButton;
        @FXML private Button updateButton;
        @FXML private Button deleteButton;
        private     final     ClientViewModel     viewModel     =
```

```
max.util.ViewModelFactory.getInstance().createClientViewModel();
        @Override protected TabId getTabId() { return
TabId.CLIENTS; }
        @Override protected Button[] getCrudButtons() {
            return new Button[]{ addButton, updateButton,
deleteButton };
        }
        /**
         * Ініціалізація контролера, налаштування прив'язок даних
та колонок таблиці.
         */
        @FXML
        public void initialize() {
            // Перевірка прав доступу (CRUD)
            applyAccessControl();
            // Налаштування колонок
            idColumn.setCellValueFactory(new
PropertyValueFactory<>("id"));
            lastNameColumn.setCellValueFactory(new
PropertyValueFactory<>("lastName"));
            firstNameColumn.setCellValueFactory(new
PropertyValueFactory<>("firstName"));
            phoneColumn.setCellValueFactory(new
PropertyValueFactory<>("phoneNumber"));
            // Прив'язка таблиці до ViewModel
            clientTable.setItems(viewModel.getClients());
            // Двосторонній біндінг полів форми
    lastNameField.textProperty().bindBidirectional(viewModel.lastN
ameProperty());
firstNameField.textProperty().bindBidirectional(viewModel.firstName
Property());
    middleNameField.textProperty().bindBidirectional(viewModel.mid
dleNameProperty());

phoneField.textProperty().bindBidirectional(viewModel.phoneProperty
());
            // Синхронізація вибраного рядка
    clientTable.getSelectionModel().selectedItemProperty().addList
ener(
                    (obs, old, newVal) ->
viewModel.selectedClientProperty().set(newVal)
            );
            viewModel.selectedClientProperty().addListener(
                    (obs, old, newVal) -> {
                        if (newVal == null) {

clientTable.getSelectionModel().clearSelection();
                        }
                    }
```

```
            );
            // Повідомлення про помилки та успіх
     errorLabel.textProperty().bind(viewModel.errorMessageProperty(
));
     successLabel.textProperty().bind(viewModel.successMessagePrope
rty());

            // Блокування кнопок під час завантаження
     addButton.disableProperty().bind(viewModel.loadingProperty());
updateButton.disableProperty().bind(viewModel.loadingProperty());
deleteButton.disableProperty().bind(viewModel.loadingProperty());
        }
        /**
         * Обробляє натискання кнопки додавання клієнта.
         */
        @FXML private void handleAddButton() { viewModel.add(); }
        /**
         * Обробляє натискання кнопки оновлення даних клієнта.
         */
        @FXML     private     void     handleUpdateButton()     {
viewModel.update(); }
        /**
         * Обробляє натискання кнопки видалення клієнта.
         */
        @FXML     private     void     handleDeleteButton()     {
viewModel.delete(); }
        /**
         * Обробляє натискання кнопки очищення полів форми.
         */
        @FXML private void handleClearButton() { viewModel.clear();
}
}
```

ДОДАТОК В – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

Позначення: Клієнт-серверний застосунок на базі архітектури REST.

1.2 Програмне забезпечення, необхідне для функціонування програми

На сервері та клієнті (клієнтах) повинна бути встановлена одна із операційних систем: Windows 10 (версії 64-bit) та вище, macOS 10.x та вище, Linux (x86-64/AArch64).

Для роботи серверної частини необхідне середовище виконання Java Runtime Environment (JRE) версії 21.

Для роботи клієнтської частини необхідне середовище виконання Java Runtime Environment (JRE) версії 21.

1.3 Мови програмування

Серверна частина застосунку «РемTech» було розроблено за допомогою мови програмування Java з використанням фреймворку Spring Boot.

Клієнтська частина застосунку була розроблена за допомогою мови програмування Java з використанням мови розмітки FXML для опису інтерфейсу користувача.

2 Функціональне призначення

Програма призначена для автоматизації таких процесів:

1. Управління базою даних клієнтів сервісного центру.

2. Облік замовлень на ремонт техніки та відстеження їх статусів.

3. Управління ремонтами: призначення майстрів, фіксація використаних запчастин та наданих послуг, відстеження статусів виконання.

4. Контроль складських запасів запчастин та прайс-листа послуг.

5. Облік даних персоналу сервісного центру.

6. Управління обліковими записами користувачів системи.

7. Формування фінансової та операційної звітності з можливістю експорту у форматах PDF, Excel (.xlsx) та CSV.

8. Розмежування прав доступу користувачів на основі рольової моделі.

Програма підтримує три ролі користувачів з різним рівнем доступу до функціональних модулів:

- Адміністратор – управляє клієнтами та замовленнями, формує акти приймання та видачі пристроїв.
- Майстер – управляє ремонтами та запчастинами, формує звіт використаних запчастин.
- Директор – управляє персоналом, послугами, обліковими записами, формує фінансові звіти.

3 Логічна структура

3.1 Алгоритми програми

Програмне забезпечення складається з клієнтської та серверної частини. Сервером є комп’ютер, на якому запущено серверну частину застосунку, клієнтською – там де запущено клієнт. Один сервер може мати багато клієнтських підключень.

Потік взаємодії частин застосунку:

1) Користувач вводить дані.

2) Клієнт перевіряє та відправляє дані на сервер.
3) Сервер перевіряє дані та взаємодіє з базою даних.
4) Сервер підтверджує успішність операції та надсилає повідомлення.
5) Клієнт відображає повідомлення про успішність операції користувачу.

3.2 Використані методи

Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп'ютерної техніки «РемTech» було розроблено на мові Java. Розробка програмного забезпечення базується на використанні об'єктно-орієнтованого підходу та окремих принципів SOLID, що сприяє підвищенню модульності, підтримуваності та гнучкості системи.

3.3 Структура програми

Програмне забезпечення складається з двох незалежних модулів, що взаємодіють через протокол HTTP у форматі JSON,

Серверна частина (Server) реалізує REST API на основі багатошарової архітектури (N-Layer):

Presentation / Controller Layer (Рівень контролерів) – обробляє HTTP-запити (GET, POST, PUT, DELETE), що надходять від клієнтської частини. Дані приймаються у форматі JSON, перетворюються у DTO та передаються до сервісного рівня.

Service Layer (Сервісний рівень) – містить бізнес-логіку застосунку. На цьому рівні реалізуються правила обробки даних, перевірка коректності операцій та координація взаємодії між компонентами.

Data Access Layer / Repository (Рівень доступу до даних) – відповідає за взаємодію з базою даних, виконання запитів та отримання або збереження даних.

Клієнтська частина (Client) реалізує десктопний інтерфейс на основі

архітектурного патерну MVVM:

Model (Модель) – рівень даних і бізнес-логіки. Містить сутності предметної області, класи доступу до даних, а також об’єкти передачі даних (DTO). Модель не має залежностей від інтерфейсу користувача.

View (Представлення) – візуальна частина застосунку, яка відповідає за відображення даних користувачеві.

ViewModel (Модель представлення) – проміжний рівень між Model і View, який забезпечує перетворення даних у формат, зручний для відображення, а також обробку дій користувача.

4 Необхідні технічні засоби

Система повинна відповідати таким мінімальним характеристикам комп’ютера:

- Процесор: двоядерний процесор Intel або AMD, з тактовою частотою 2 ГГц.
- Оперативна пам’ять: 4 ГБ.
- Дисковий простір: HDD/SSD на 128 ГБ.
- Мережеві можливості: мережевий адаптер з підтримкою Ethernet для дротової локальної мережі, або підтримка WIFI для бездротової локальної мережі.

5 Виклик і запуск

Запуск серверної частини. Виконується запуск виконуваного файлу Server.jar. Сервер за замовчуванням прослуховує порт 8080. При першому запуску автоматично створюється файл бази даних servicedb.mv.db та обліковий запис адміністратора з логіном admin і паролем admin.

Запуск клієнтської частини. Виконується запуск файлу Client.jar. При запуску відображається вікно авторизації. Для успішного входу сервер повинен

бути запущений та доступний по мережі. Якщо з'єднання з сервером відсутнє, після введення облікових даних відображається повідомлення про помилку мережі.

Конфігурація мережевої адреси. Адреса сервера задається у файлі config.properties, що знаходиться в тій самій директорії, що й Client.jar. Параметр: «server.url=http://XXX.XXX.X.X:8080». Адресу комп'ютера-сервера можно дізнатися за допомогою команди «ifconfig»/«ip» у термінали системи. Якщо файл відсутній, клієнт автоматично підключається до http://localhost:8080.

6 Вхідні та вихідні дані

Вхідні дані: дані введені користувачем, що вводяться через форми JavaFX (текстові поля, випадаючі списки, селектори дат), які передаються на сервер у форматі JSON.

Вихідні дані: Інтерактивні таблиці з результатами вибірки з бази даних.

Згенеровані звіти у форматах .pdf (бібліотека OpenPDF), .xlsx (Apache POI) та .csv.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

1 Призначення та умови застосування

Програма «РемTech» призначена для автоматизації процесів обліку замовлень, контролю складських запасів та формування звітності в сервісних центрах з ремонту комп'ютерної техніки.

Для роботи системи необхідно, щоб серверна частина була запущена на виділеному комп'ютері (або на тому самому комп'ютері, що й клієнт) та доступна по локальній мережі. Клієнтський застосунок підключається до сервера за адресою, вказаною у файлі config.properties.

Авторизація в системі вимагає наявності облікового запису (логін та пароль). За замовчуванням при першому запуску сервера автоматично створюється обліковий запис з логіном admin і паролем admin та роллю «Директор».

Рекомендація безпеки. Після першого входу рекомендується змінити пароль облікового запису за замовчуванням. Для цього перейдіть на вкладку «Профілі», оберіть запис admin, введіть новий пароль у поле «Пароль» та натисніть «Зберегти».

2 Початок роботи

Авторизація: Запустіть клієнтський застосунок. У вікні «Авторизація» введіть логін та пароль і натисніть кнопку «Увійти» (рисунок Г.1). При першому запуску використовуйте дані: логін – admin, пароль – admin.

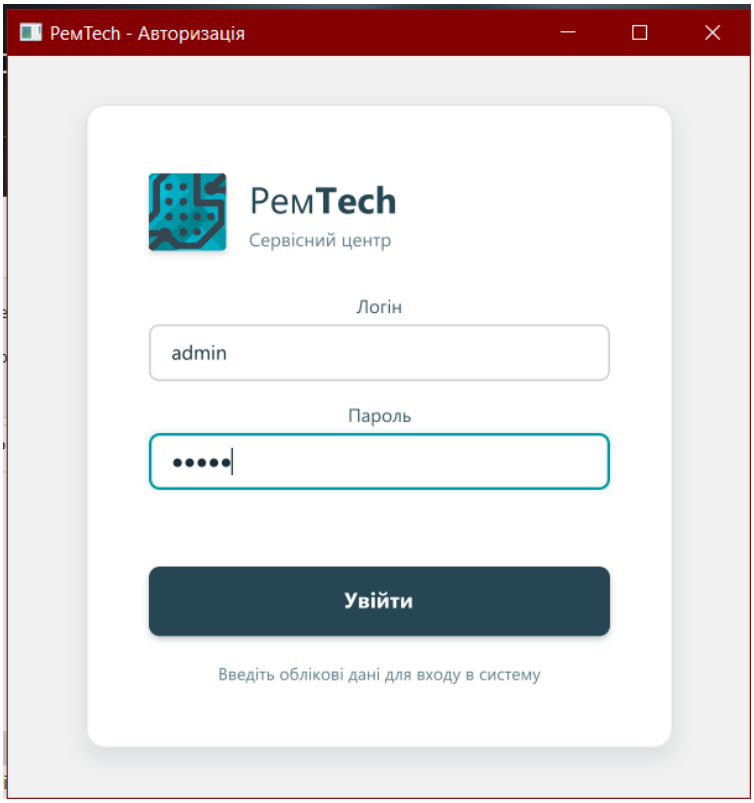


Рисунок Г.1 – Авторизація

Якщо з'явилося повідомлення про помилку входу, перевірте:

- чи запущена серверна частина (Server.jar);
- чи правильно вказана адреса сервера у файлі config.properties;
- чи введені коректні облікові дані.

Головне вікно: Після успішного входу відкривається головне вікно із вкладками. Перелік доступних вкладок залежить від ролі облікового запису (Майстер, Адміністратор або Директор) (рисунок Г.2).

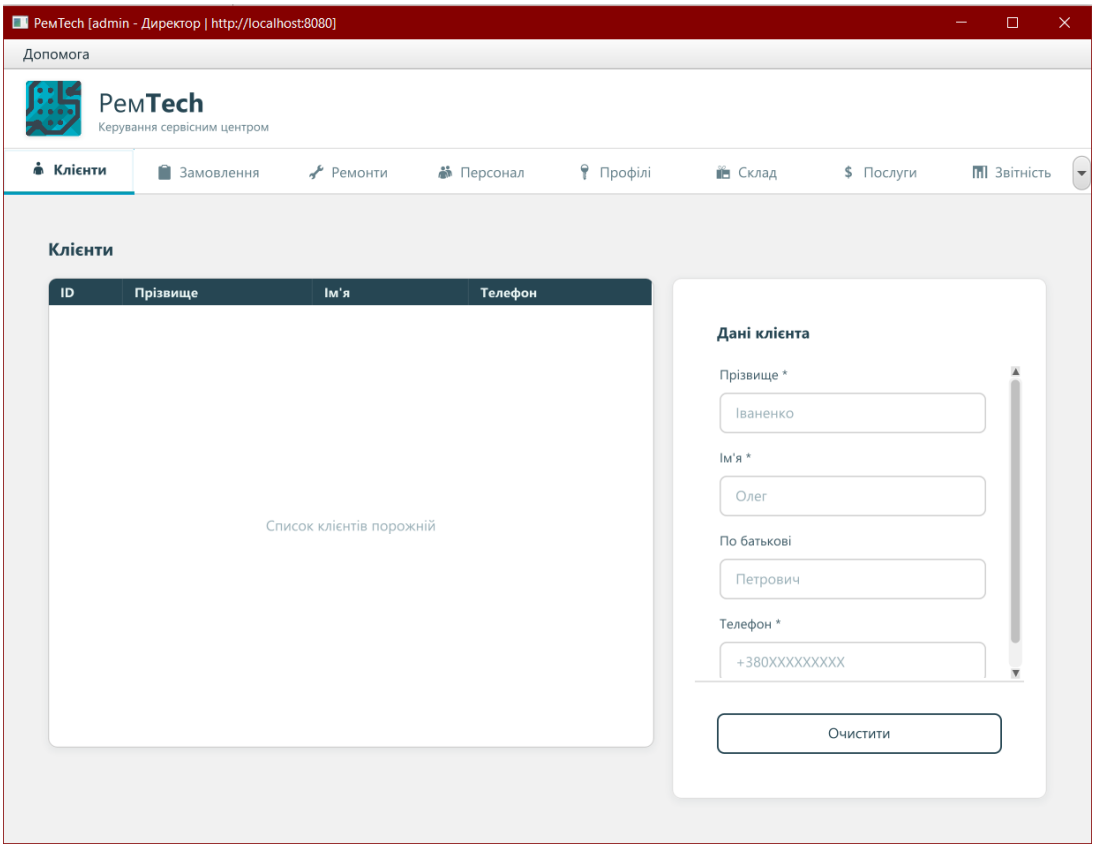


Рисунок Г.2 – Головне вікно

3 Керування записами таблиць

3.1 Робота із записами.

Перегляд: Всі сутності системи відображаються у вигляді таблиць на відповідних вкладках. Інтерфейс кожної вкладки складається з таблиці зліва та панелі форми справа.

Перегляд: Усі записи відображаються у таблиці на відповідній вкладці.

Додавання: Для додавання заповніть форму праворуч та натисніть «Додати».

Редагування: Оберіть запис у таблиці, змініть дані у відповідних полях та натисніть «Оновити».

Видалення: Оберіть запис у таблиці, змініть дані у відповідних полях та натисніть «Видалити».

На рисунку Г.3 представлено всі елементи для роботи із записами: вкладки,

таблицю, форми, кнопки.

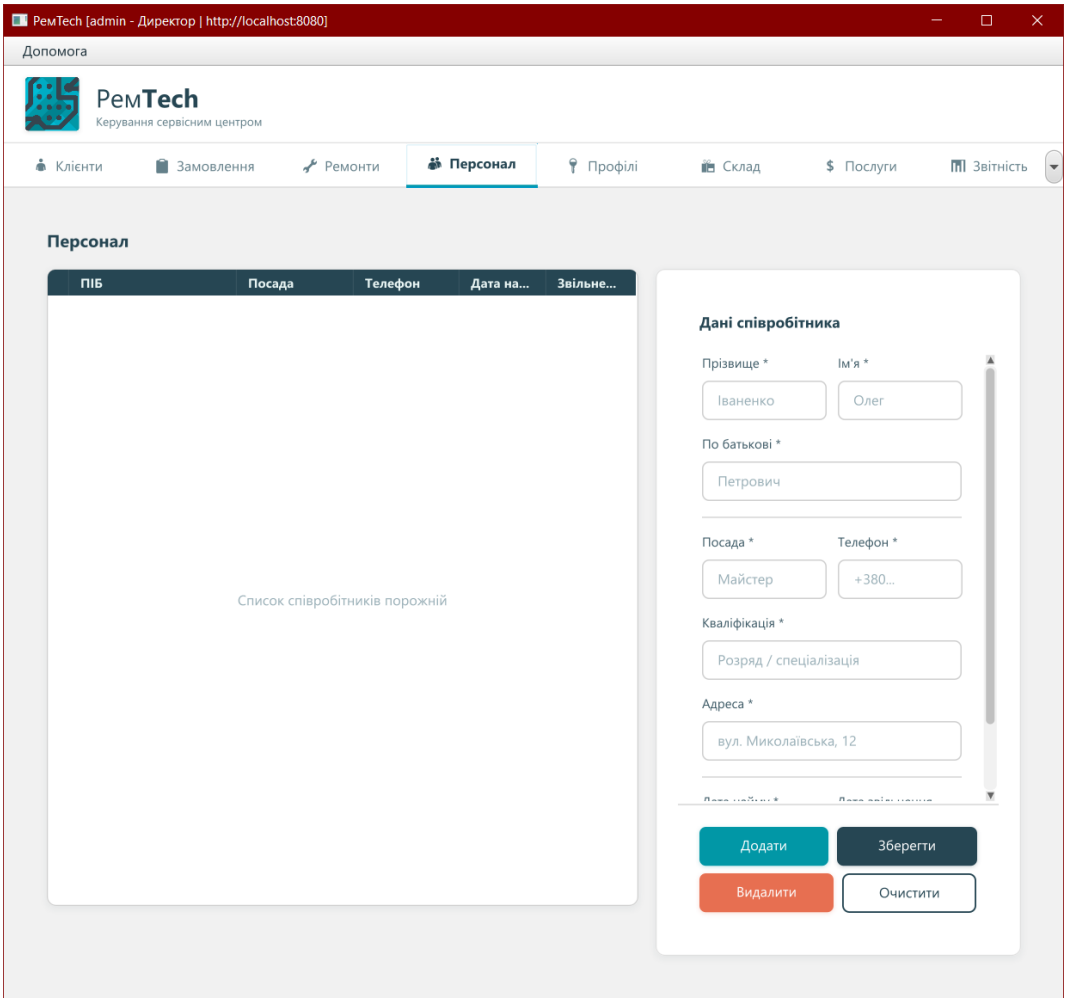


Рисунок Г.3 – Взаємодія із записами

3.2 Робота із замовленнями та ремонтами

Прийом пристрою: Перейдіть на вкладку «Замовлення», заповніть всі обов’язкові поля у формі: оберіть клієнта, вкажіть назву пристрою, модель та опис проблеми, дату прийому та орієнтовну дату видачі. Натисніть «Додати» (рисунок Г.4).

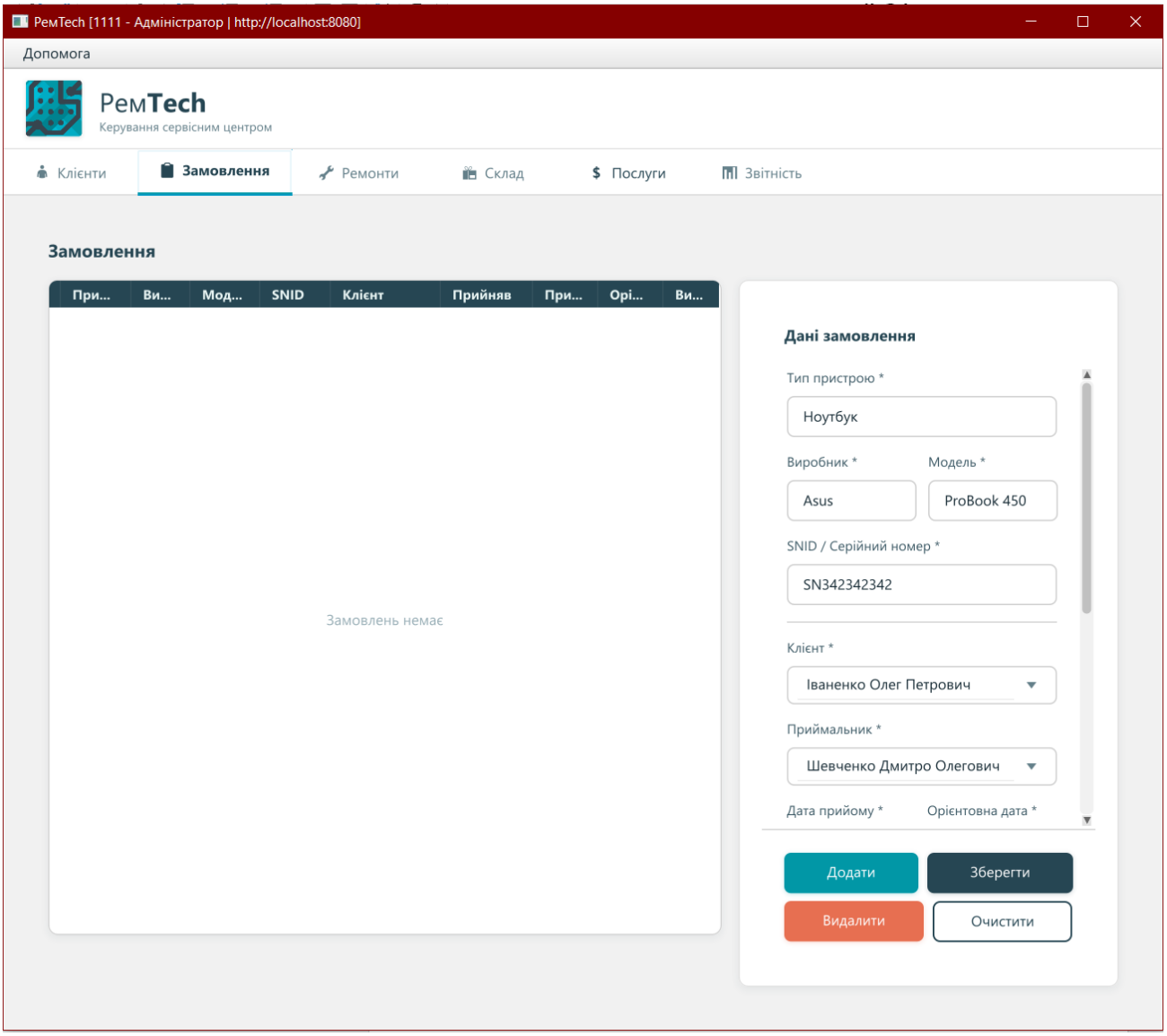


Рисунок Г.4 – Прийом пристрою

Виконання ремонту: На вкладці «Ремонт», заповніть всі обов’язкові поля у формі: оберіть замовлення, статус, дату прийому та майстра (майстрів), також протягом ремонту можна заповнити опціональні поля: використані запчастини та надані послуги. Застосунок автоматично розрахує вартість ремонту на основі доданих запчастин та послуг (рисунок Г.5).

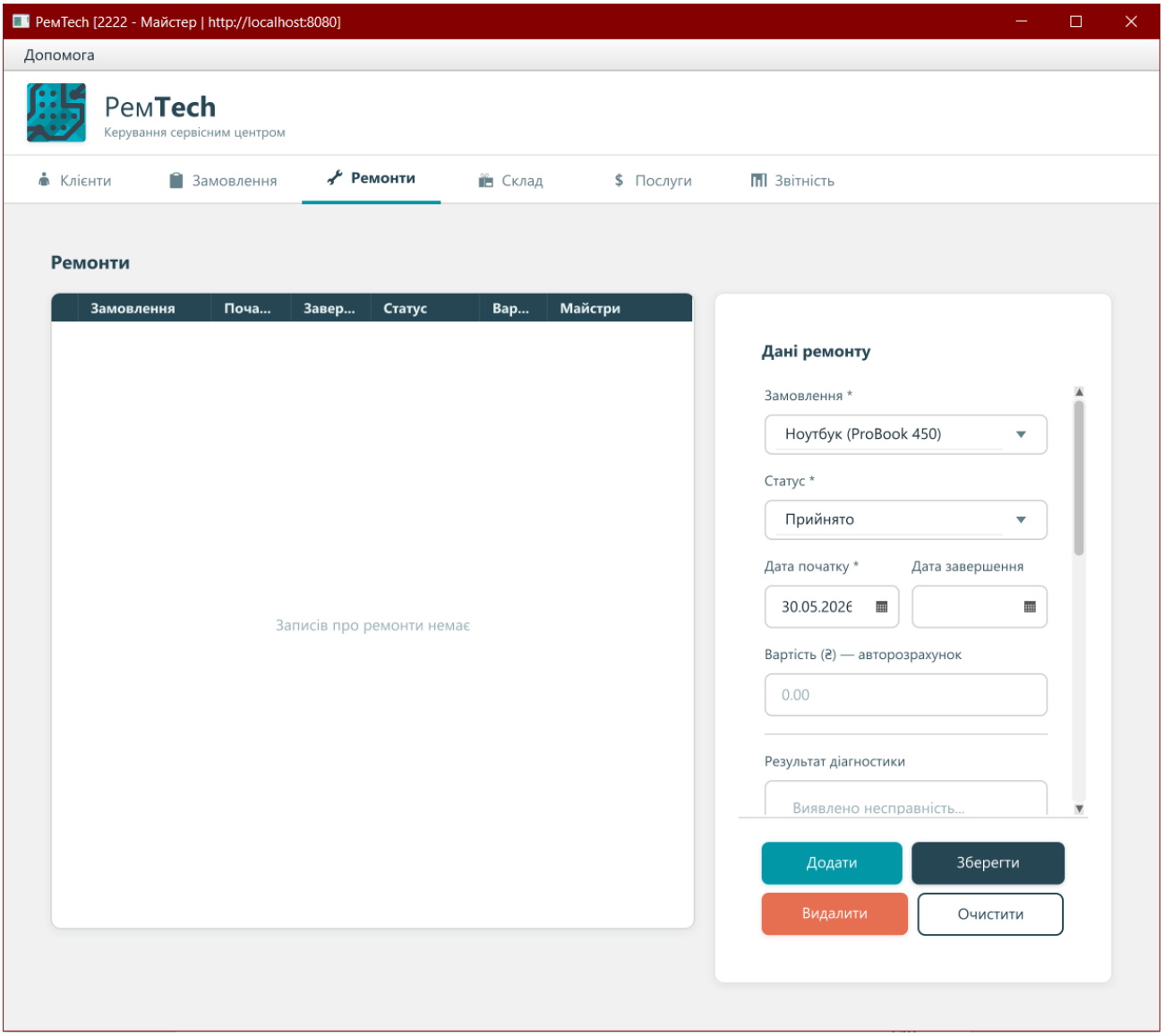


Рисунок Г.5 – Виконання ремонту

Зміна статусу: Виберіть запис у таблиці «Ремонти», змініть статус у випадаючому списку («Прийнято», «Діагностика», «Виконано», тощо) та натисніть «Оновити» (рисунок Г.6).

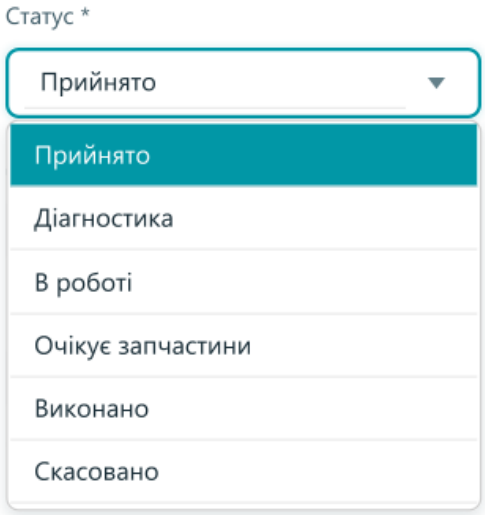


Рисунок Г.6 –Зміна статусу ремонту

Закінчення ремонту та закриття замовлення: На вкладці «Ремонт» для існуючого ремонту потрібно внести такі дані: статус «Виконано» або «Скасовано» та дату закінчення. На вкладці «Замовлення» для існуючого замовлення додати дату фактичної видачі.

3.3 Склад та послуги

Запчастини: Вкладка «Склад» дозволяє вести облік кількості деталей та їхньої вартості. Додані запчастини можуть бути додані до ремонтів, та будуть відображатись у звітах (рисунок Г.7).

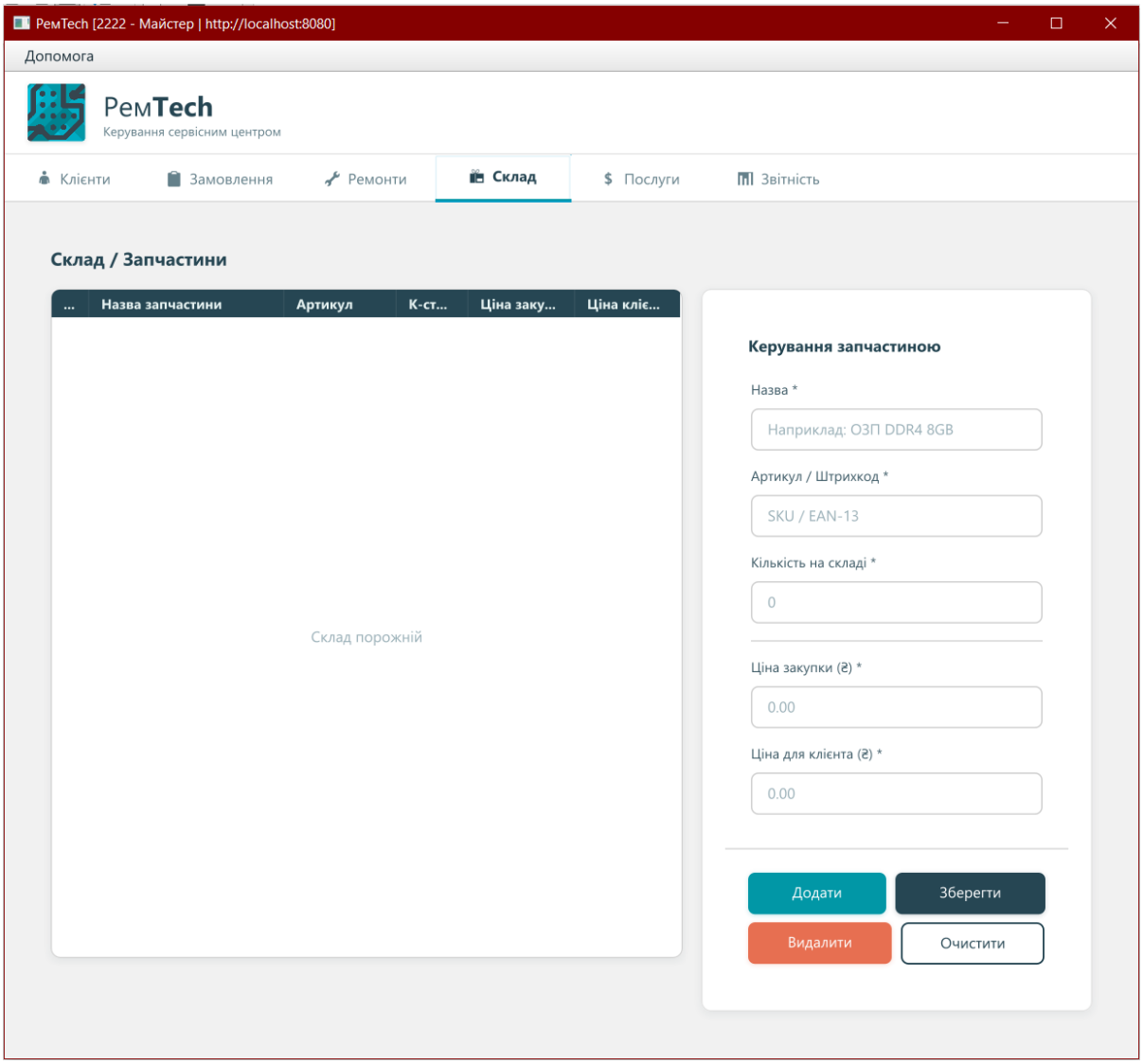


Рисунок Г.7 – Вкладка «Склад»

Прайс-лист: На вкладці «Послуги» встановлюються фіксовані ціни на типи робіт. Додані послуги можуть бути додані до ремонтів, та будуть відображатись у звітах (рисунок Г.8).

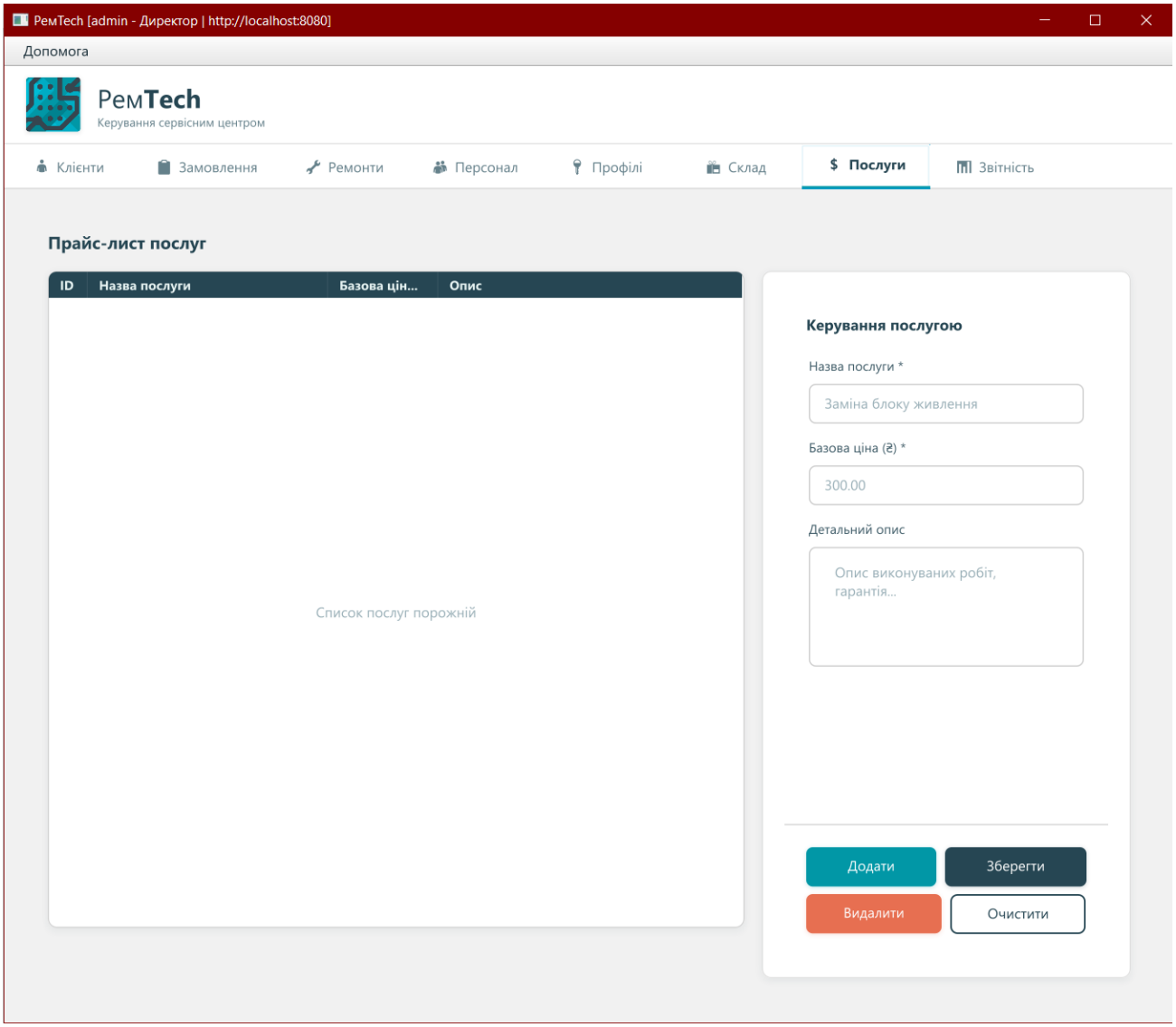


Рисунок Г.8 – Вкладка «Послуги»

3.4 Генерація звітів

Перейдіть до вкладки «Звітність», оберіть потрібний звіт, та заповніть форму:

1. Акт приймання: Цей документ формується для створеного запису таблиці «Замовлення» у момент приймання пристрою працівником сервісного центру.

2. Акт видачі: Цей документ формується для закритого замовлення та завершеного ремонту цього замовлення при видачі пристрою власнику

робітником сервісного центру.

3. Фінансовий звіт: Цей документ формується для закритих замовлень із завершеним ремонтом.

4. Використані запчастини: Цей документ формується для завершених ремонтів.

Після цього буде доступне формування звіту у застосунку, натисніть «Сформувати звіт». Після створення звіту, буде можливість зберегти у форматах відповідними кнопками «XLSX», «CSV», «PDF».

На рисунку Г.9 представлено всі елементи для роботи зі звітами.

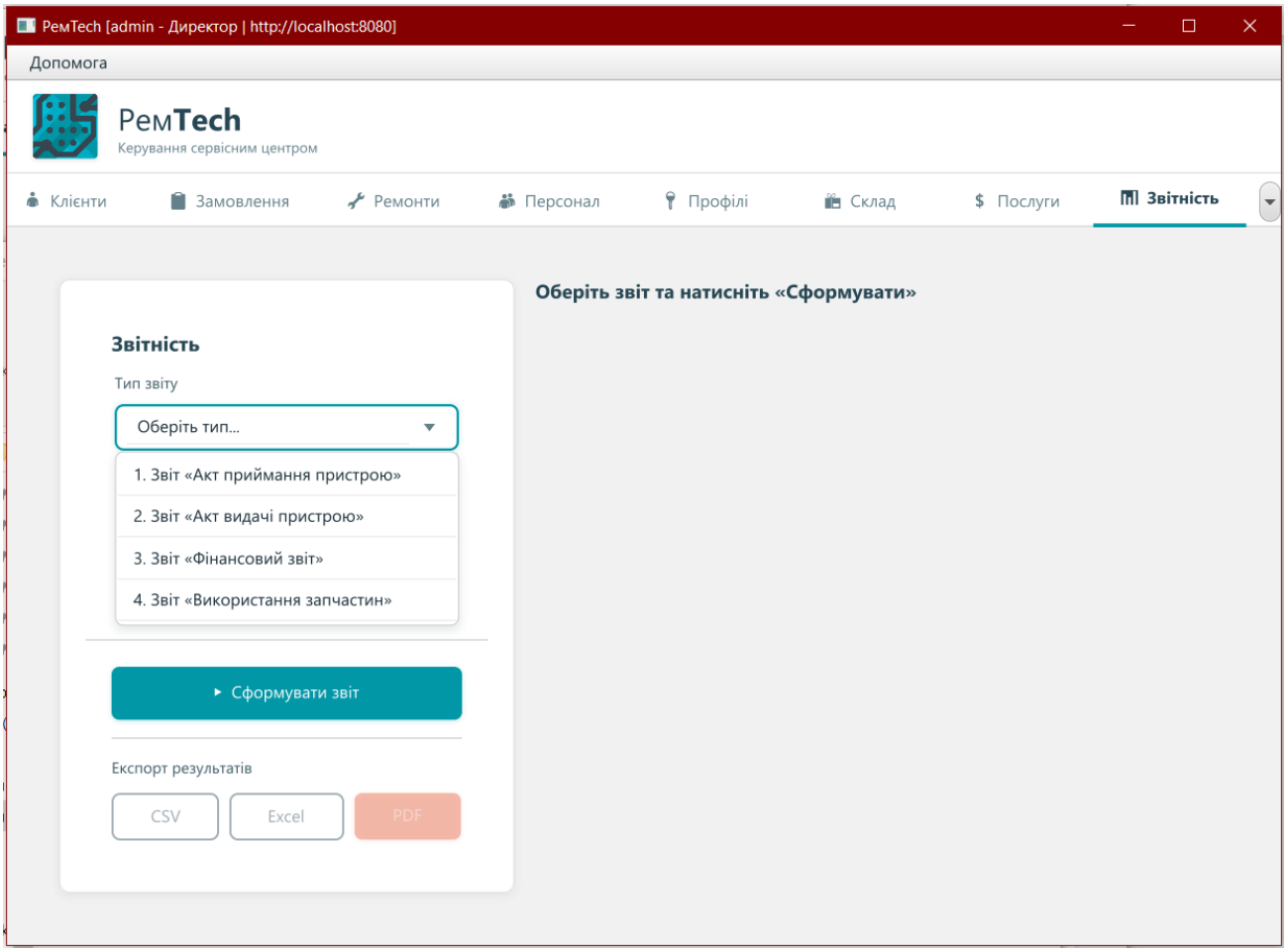


Рисунок Г.9 – Звітність

3.5 Облікові записи

Вкладка «Профілі» надає функціонал управління обліковими записами. Робота з записами здійснюється за загальним принципом, описаним у розділі 3.1. Облікові записи використовуються для авторизації з відповідною роллю. При створенні нового облікового запису обов'язково вкажіть логін, пароль та роль (Майстер, Адміністратор або Директор) (рисунок Г.10).

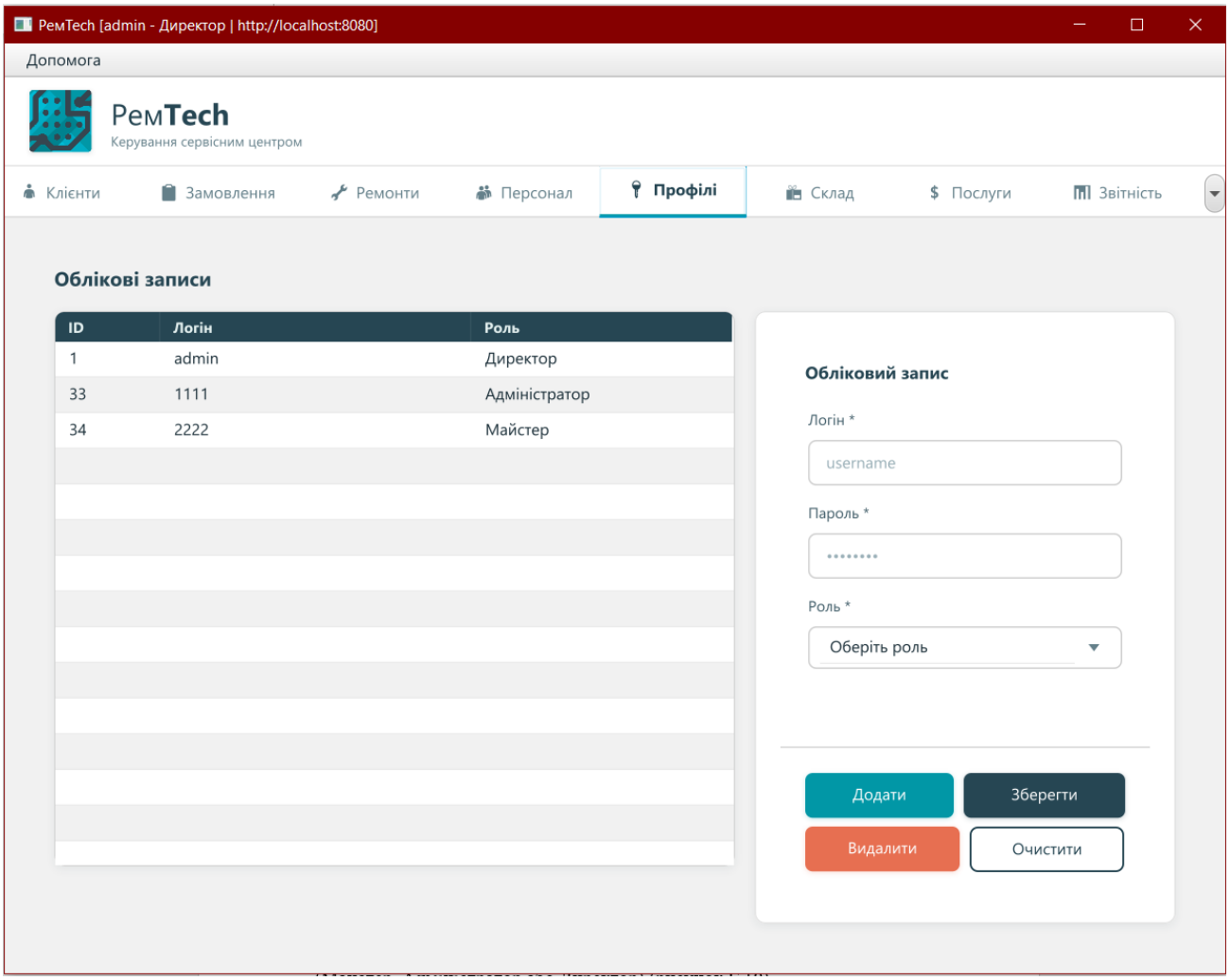


Рисунок Г.10 – Вкладка «Профілі»

4 Можливі проблеми та їх вирішення, допомога

Можливі проблеми та вирішення представлено у таблиці Г.1.

Таблиця Г.1 – Можливі проблеми та вирішення

Симптом	Можлива причина	Дія
Повідомлення «Помилка входу» при авторизації	Не запущена серверна частина або неправильна адреса	Запустити Server.jar; перевірити config.properties
Повідомлення «Невірний пароль»	Введено неправильні облікові дані	Перевірити логін та пароль
Кнопки «Додати» / «Зберегти» / «Видалити» відсутні	Недостатньо прав доступу для поточної ролі	Переглянути таблицю розподілу ролей
Таблиця порожня після відкриття вкладки	Ще немає записів або втрачено з'єднання з сервером	Перевірити наявність даних; перезапустити клієнт
Звіт порожній	Немає даних за вказаний діапазон дат або ID	Перевірити коректність введених параметрів

У застосунку наявне вікно «Допомога», де користувач може знайти додаткову інформацію чи ознайомитись із застосунком (рисунок Г. 11).

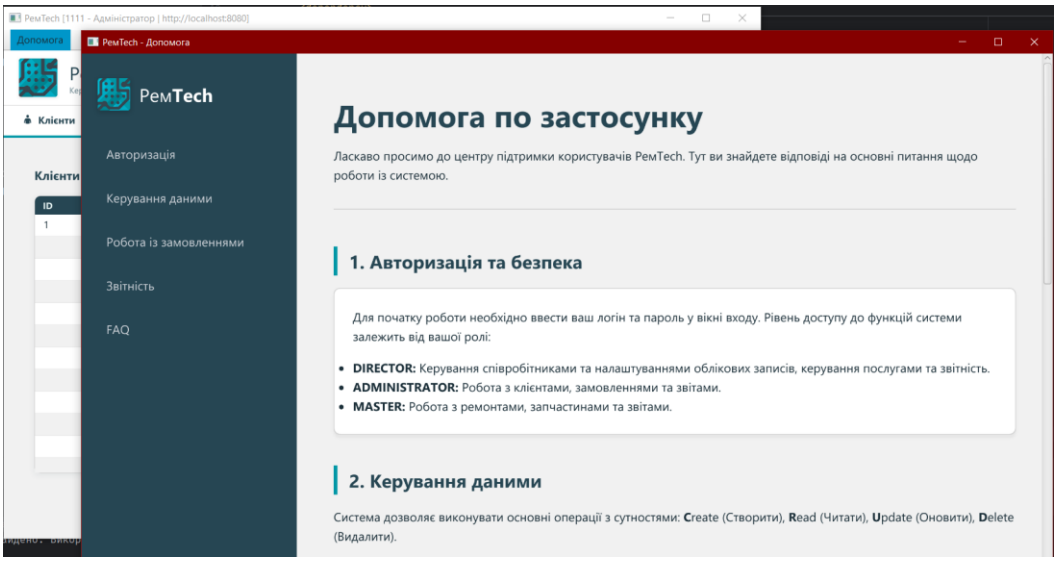


Рисунок Г.11 – Допомога

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕМТЕСН»

1 Об’єкт випробувань

1.1 Найменування випробуваної програми

Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

1.2 Область застосування випробуваної програми

Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» призначено для автоматизації повного циклу обслуговування клієнтів – від первинної реєстрації несправного пристрою та клієнта до формування звітності, з метою покращення швидкості та якості обслуговування, спрощення та автоматизації роботи працівників, та вдосконалення досвіду клієнта при відвідуванні сервісного центра.

1.3 Позначення випробуваної програми

«Програмне забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech» розроблено у межах кваліфікаційної роботи «Розробка програмного забезпечення для автоматизації обліку виконання робіт сервісного центру з ремонту комп’ютерної техніки «РемTech».

2 Мета випробувань

Метою випробувань є перевірка працездатності та відповідності розробленого застосунку характеристикам та вимогам, які викладені у технічному завданні та наведено у додатку А.

3 Вимоги до програми

При проведені випробувань перевіряються функціональні характеристики, застосунок, підлягає перевірці на відповідність вимогам технічного завдання, які викладені у пункті «Вимоги до функціональних характеристик» у технічному завданні – додаток А.

4 Вимоги до програмної документації

До складу програмної документації повинні входити наступні документи:

- технічне завдання (Додаток А);
- текст програми (Додаток Б);
- опис програми (Додаток В);
- інструкція користувача (Додаток Г);
- програма та методика випробувань (Додаток Д).

5 Засоби і порядок випробувань

5.1 Склад технічних засобів, що використовуються під час випробувань

Склад технічних засобів що використовувались під час випробувань:

- Основна робоча станція: процесор 14 ядер, 4.1 ГГц; оперативна пам’ять 32 ГБ; SSD 300 ГБ вільного місця, мережевий адаптер з підтримкою проводового та бездротового підключення 1000 Мбіт/с, операційна система Windows 10 21H2 (64-bit).
- Віртуальна машина – клієнт №2: Oracle VM VirtualBox; операційна система Windows 10 22H2 (64-bit); 4 ГБ оперативної пам'яті, 2 ядра.
- Віртуальна машина – клієнт №3: Oracle VM VirtualBox; операційна система Linux Mint 22.3 (x86-64); 4 ГБ оперативної пам'яті, 2 ядра.
- Локальна мережа: пропускна здатність 1000 Мбіт/с (Ethernet).

Всі три конфігурації відповідають мінімальним системним вимогам, визначеним у технічному завданні.

5.2 Програмні засоби, що використовуються під час випробувань

При проведені випробувань використовуються такі програмні засоби:

- Java Runtime Environment (JRE) версії 21 – середовище виконання для серверної та клієнтської частин.
- Операційні системи: Windows 10 21H2, Windows 10 22H2, Linux Mint 22.3 – для перевірки кросплатформності.
- Oracle VM VirtualBox – для розгортання клієнтських вузлів на віртуальних машинах.
- Apache JMeter 5.6 – для навантажувального тестування REST API (перевірка стабільності сервера при кількох одночасних підключеннях).
- Браузер (H2 Console) – для безпосередньої перевірки стану бази даних після операцій (доступна за адресою http://localhost:8080/h2-console).

5.3 Порядок проведення випробувань

5.3.1 Підготовка до випробувань

Перед початком випробувань виконуються такі підготовчі дії:

1. На сервері запускається серверна частина командою java -jar Server.jar. Сервер повинен успішно стартувати на порту 8080 та автоматично створити файл бази даних servicedb.mv.db і обліковий запис admin / admin з роллю «Директор».

2. Файл config.properties з рядком server.url=http://localhost:8080 розміщується поруч із Client.jar.

3. На всіх клієнтських вузлах запускається клієнтська частина командою java -jar Client.jar.

4. Перевіряється відображення вікна авторизації на кожному вузлі.

5. Для тестування ролей заздалегідь через вкладку «Профілі» (під обліковим записом директора) створюються облікові записи:

- admin_test / test123 – роль Адміністратор;
- master_test / test123 – роль Майстер.

6. Заздалегідь вносяться початкові дані: щонайменше 2 клієнти, 2 співробітники, 2 замовлення, 2 ремонти, 2 запчастини, 2 послуги.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Основним методом тестування є тестування «чорної скриньки» методом класів еквівалентності (Equivalence Class Partitioning). Для кожного поля вводу виділяються правильні класи еквівалентності (ПКЕ – значення, що мають прийматися системою) та неправильні класи еквівалентності (НКЕ – значення, що мають відхилятися з відповідним повідомленням). Детальні таблиці класів еквівалентності наведено у підрозділі 2.3.3 основного тексту кваліфікаційної роботи.

Додатково застосовуються:

- Тестування граничних значень – перевірка мінімально та максимально допустимих значень полів (наприклад, телефон від 10 до 13 цифр).
- Функціональне тестування рольової моделі – перевірка відображення вкладок і доступності кнопок для кожної з трьох ролей.
- Інтеграційне тестування – перевірка взаємодії клієнтської та серверної частин через REST API.
- Ручне системне тестування – відтворення реальних сценаріїв роботи персоналу (прийом замовлення -> призначення ремонту -> зміна статусу -> формування акту видачі).

Через велику кількість функцій, які треба випробовувати, буде представлено лише декілька результатів випробувань

6.2 Програма випробувань

6.2.1 Випробування функції «Авторизація» (В-01)

Тест А-01.1 – Успішна авторизація.

Умова: у систему введено коректний логін і пароль існуючого облікового запису.

Послідовність дій:

1. Відкрити клієнтський застосунок.

2. Ввести логін admin та пароль admin.

3. Натиснути «Увійти».

Очікуваний результат: відкривається головне вікно з вкладками відповідно до ролі «Директор». Рядок заголовку вікна містить логін і роль поточного користувача.

Тест А-01.2 – Авторизація з невірним паролем.

Послідовність дій:

1. Ввести коректний логін admin, пароль wrongpass.

2. Натиснути «Увійти».

Очікуваний результат: відображається повідомлення про помилку. Головне вікно не відкривається. Поля введення залишаються активними.

Тест А-01.3 – Авторизація без запущеного сервера.

Послідовність дій:

1. Зупинити серверну частину.

2. Спробувати увійти з коректними даними.

Очікуваний результат: відображається повідомлення про помилку мережі.

Тест А-01.4 – Перевірка відображення вкладок за роллю.

Послідовність дій:

1. Виконати вхід з обліковим записом master_test (роль Майстер).

2. Перевірити наявні вкладки.

Очікуваний результат: вкладки «Персонал» і «Профілі» відсутні. Присутні

«Клієнти», «Замовлення», «Ремонти», «Склад», «Послуги», «Звітність». Кнопки «Додати», «Зберегти», «Видалити» відсутні на вкладках «Клієнти», «Замовлення», «Послуги».

6.2.2 Випробування функцій управління клієнтами (В-02)

Тест В-02.1 – Додавання клієнта з коректними даними (ПКЕ).

Послідовність дій:

1. Перейти на вкладку «Клієнти».

2. Заповнити поля: Прізвище – «Іваненко», Ім'я – «Олег», Телефон – «+380501234567».

3. Натиснути «Додати».

Очікуваний результат: запис з'явився в таблиці. Відображається повідомлення про успіх. Поля форми очищені.

Тест В-02.2 – Додавання клієнта з порожнім прізвищем (НКЕ).

Послідовність дій:

1. Залишити поле «Прізвище» порожнім, заповнити інші поля коректно.

2. Натиснути «Додати».

Очікуваний результат: відображається повідомлення «Прізвище є обов'язковим». Запис у таблицю не додається.

Тест В-02.3 – Додавання клієнта з некоректним форматом телефону (НКЕ).

Послідовність дій:

1. Ввести телефон abc123.

2. Натиснути «Додати».

Очікуваний результат: відображається повідомлення про некоректний формат номера телефону.

Тест В-02.4 – Редагування клієнта.

Послідовність дій:

1. Обрати існуючий запис у таблиці.

2. Змінити прізвище.

3. Натиснути «Зберегти».

Очікуваний результат: запис у таблиці оновлено. Відображається повідомлення про успіх.

Тест В-02.5 – Видалення клієнта.

Послідовність дій:

1. Обрати існуючий запис.

2. Натиснути «Видалити».

Очікуваний результат: запис зникає з таблиці.

Тест В-02.6 – Перевірка обмежень доступу для ролі Майстер.

Послідовність дій:

1. Увійти як master_test.

2. Перейти на вкладку «Клієнти».

Очікуваний результат: кнопки «Додати», «Зберегти», «Видалити» відсутні. Таблиця доступна лише для перегляду.

6.2.3 Випробування функцій управління замовленнями (В-03)

Тест В-03.1 – Додавання замовлення з коректними даними (ПКЕ).

Послідовність дій:

1. Перейти на вкладку «Замовлення».

2. Заповнити всі обов'язкові поля: тип пристрою, виробник, модель, SNID, опис стану, опис проблеми; обрати клієнта та приймальника; вказати дату прийому та орієнтовну дату.

3. Натиснути «Додати».

Очікуваний результат: замовлення з'являється в таблиці.

Тест В-03.2 – Додавання замовлення без клієнта (НКЕ).

Послідовність дій:

1. Заповнити всі поля, крім поля «Клієнт».

2. Натиснути «Додати».

Очікуваний результат: відображається повідомлення «Виберіть клієнта».

Тест В-03.3 – Додавання замовлення з орієнтовною датою, що раніше дати прийому (НКЕ).

Послідовність дій:

1. Встановити дату прийому 01.05.2026, орієнтовну дату 01.04.2026.

2. Натиснути «Додати».

Очікуваний результат: відображається повідомлення про некоректний діапазон дат.

6.3 Випробування нефункціональних вимог

Тест НФ-01 – Кросплатформність.

Послідовність дій:

1. Запустити клієнтський застосунок на Windows 10, Windows 10 (VM), Linux Mint 22.3 (VM), MacOS.

2. На кожному вузлі виконати авторизацію та будь-яку CRUD-операцію.

Очікуваний результат: застосунок запускається та коректно функціонує на всіх трьох операційних системах.

Тест НФ-02 – Одночасна робота кількох клієнтів.

Послідовність дій:

1. Увійти в систему одночасно з трьох клієнтських вузлів.

2. З першого вузла додати клієнта.

3. Протягом 5 секунд перевірити відображення нового запису на другому та третьому вузлах.

Очікуваний результат: завдяки механізму автоматичного оновлення, новий запис відображається на всіх вузлах не пізніше ніж через 3-6 секунд.

Тест НФ-03 – Перевірка конфігурації мережі через config.properties.

Послідовність дій:

1. Змінити значення server.url у файлі config.properties клієнта на адресу іншого вузла в локальній мережі.

2. Запустити клієнт та виконати авторизацію.

Очікуваний результат: клієнт підключається до вказаного сервера без перекомпіляції.

Тест НФ-04 – Автоматичне створення бази даних при першому запуску.

Послідовність дій:

1. Видалити файли servicedb.mv.db та servicedb.trace.db.

2. Запустити Server.jar.

Очікуваний результат: файли бази даних та обліковий запис admin / admin створені автоматично.

Тест НФ-05 – Навантажувальне тестування REST API за допомогою Apache JMeter.

Мета: перевірити стабільність серверної частини при одночасному надходженні кількох HTTP-запитів і переконатися, що сервер обробляє їх коректно без помилок і зависань.

Передумови: сервер запущений (Server.jar, порт 8080); Apache JMeter 5.6 встановлений та запущений; у базі даних є щонайменше 5 записів клієнтів.

Послідовність дій:

1. Запустити Apache JMeter. Створити новий тестовий план: File -> New.

2. Додати Thread Group: права кнопка на Test Plan -> Add -> Threads -> Thread Group. Встановити: Number of Threads – 10; Ramp-Up Period – 3 с; Loop Count – 30.

3. Додати HTTP Request: права кнопка на Thread Group -> Add -> Sampler -> HTTP Request. Встановити: Protocol – http; Server Name – localhost; Port – 8080; Method – GET; Path – /api/clients.

4. Додати HTTP Header Manager: Add -> Config Element -> HTTP Header Manager. Додати заголовок Content-Type: application/json.

5. Додати View Results Tree та Summary Report: Add -> Listener -> відповідний пункт.

6. Натиснути кнопку «Запустити» (зелена стрілка або Ctrl+R).

Очікуваний результат:

- Усі 300 запитів (10 потоків * 30 ітерації) завершуються зі статус-кодом HTTP 200.
- Показник Error % у Summary Report дорівнює 0%.
- Середній час відповіді (Average) не перевищує 500 мс у локальній мережі.
- Сервер залишається працездатним і продовжує відповідати на запити після завершення тесту.