НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____комп'ютерних наук_____
(назва кафедри)

_____ ____Голуб Белла Львівна___/
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Інформаційна система обліку обслуговування клієнтів компаній ІТ»

Спеціальність 122 — «Комп'ютерні науки»
ОП «Комп'ютерні науки»

Гарант освітньої програми

д.е.н., професор		Руденський Р.А.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.т.н., доцент		Голуб Б.Л.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав _____ Полянський Богдан Вадимович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____к.т.н., доцент_____ _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"_16_" __грудня__ 2024 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Полянському Богдану Вадимовичу

Спеціальність 122 — «Комп'ютерні науки»

ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система обліку обслуговування клієнтів компаній ІТ» затверджена наказом ректора НУБіП України від 16.12.2024 № _2246_С_

2. Термін подання завершеної роботи на кафедру____2025.05.25____
(рік, місяць, число)

3. Вихідні дані до бакалаврської кваліфікаційної роботи: документації CRM систем.

4. Перелік питань, що розглядаються:

- Системний аналіз предметної області обслуговування клієнтів компаній ІТ.
- Інформаційне забезпечення.
- Прикладне програмне забезпечення.
- Рекомендації щодо впровадження та експлуатації системи.

Дата видачі завдання "_16_"_____грудня_____ 2024 р.

Керівник бакалаврської кваліфікаційної роботи _____ Голуб Б.Л._____
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Полянський Б.В._____/
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних.

ІС — інформаційна система.

ПЗ — програмне забезпечення.

СУБД — система управління базами даних.

CRM (Customer Relationship Management) — система керування взаєминами з клієнтами, яка допомагає компаніям організовувати, аналізувати та покращувати взаємодію на всіх етапах співпраці.

FK (Foreign Key) — зовнішній ключ, що встановлює зв’язок між таблицями в базі даних.

ID (Identifier) — унікальний ідентифікатор запису, який зазвичай використовується як первинний ключ у таблиці бази даних.

SQL (Structured Query Language) — мова структурованих запитів.

ВСТУП

Актуальність розробки. Нині, в епоху диджиталізації, світ суттєво змінюється, внаслідок чого різні сфери, такі як бізнес, освіта, переходять на інформаційні системи, CRM та інші. Сфера обслуговування клієнтів не є виключенням. Використовуючи старі методи, є багато різних неоптимальних процесів, які бажано автоматизувати. Також є потреба у швидкій обробці інформації, яку може забезпечити ІС, аналізуючи дані та формуючи необхідну вихідну інформацію.

Раніше для виклику працівника служби підтримки клієнту було потрібно дізнатися їх номер, зателефонувати на нього, передати їм свої контактні дані та повідомити про технічну проблему, обрати дату та час, після чого вони шукають в списку вільного на той час працівника у потрібному місті та призначають його на заявку візиту, а потім, коли настає час візиту, потрібно уточнювати у клієнта чи він буде присутнім, чи потрібно скасувати його заявку.

Тому необхідною є система для подачі клієнтами заявок на візит у реальному часі, автоматичного створення журналу заявок на зустріч, заповнення працівниками актів виконаних робіт для ведення звітності компанії. Таким чином IT-сервіси обслуговування клієнтів потребують інструментів для автоматичного планування, обліку та контролю виконаних робіт для підвищення ефективності їх роботи та обробки великих обсягів даних, автоматизації процесів, таких як ведення звітності, створення журналу заявок та їх подачі клієнтами.

Мета розробки. Мета розробки — автоматизація процесів, пов’язаних з обслуговуванням клієнтів за допомогою створення інформаційної системи обліку обслуговування клієнтів компаній IT.

Методи та технології. Розробка програмного забезпечення здійснювалася з використанням таких технологій:

- C# (мова програмування для створення логіки застосунку);
- Windows Forms (API, який забезпечує функціонування та взаємодію графічного інтерфейсу користувача у програмних додатках);
- Microsoft SQL Server (система керування базами даних для зберігання та обробки даних);
- SQL Server Management Studio (інструмент для розгортання та адміністрування бази даних);
- ADO.NET (технологія для підключення до бази даних і виконання SQL-запитів);
- Google Cloud SQL (хмарна платформа для зберігання бази даних та її резервної копії, забезпечення віддаленого доступу).

Апробація програмного додатку. Результати дипломної роботи були апробовані під час участі в міжнародній конференції: **«День науки і 100-ліття академіка Е.Букетова: III Міжнародна науково-теоретична конференція студентів»**, яка відбулася в університеті дружби народів імені академіка А.Куатбекова (Республіка Казахстан).

Також було опубліковано результати дипломної роботи у тезах виступу на VII Всеукраїнській науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем ‘2025’»: **Полянський Б.В.** Інформаційна система обліку обслуговування клієнтів компаній IT / Тези доповіді на студентській конференції «Теоретичні та прикладні аспекти розробки комп’ютерних систем ‘2025’». — Київ, 2025.

Структура записки. Дипломна записка складається з 65 сторінок, включає 23 використаних джерела та п’ять додатків, основна частина розділена на такі розділи:

1) Системний аналіз предметної області, в якому виконано моделювання предметної області, використовуючи UML діаграми, розглянуто існуючі рішення та описано постановку завдання.
2) Інформаційне забезпечення, у якому розроблено логічну модель даних, обрано систему управління інформаційною базою та створено її для зберігання та обробки необхідної інформації компаній, їх працівників та клієнтів.
3) Прикладне програмне забезпечення, що містить опис програмного додатка, його організаційної структури, обрані інструменти для розробки, проектування і реалізацію взаємодії з базою даних і сервісами.
4) Рекомендації щодо впровадження та експлуатації системи, в якому описано тестування програмного застосунку, визначено вимоги до апаратного та програмного забезпечення комп’ютерів користувачів та сервера, сформовано інсталяційний пакет для встановлення необхідних файлів для роботи програми на комп’ютерах користувачів.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОБСЛУГОВУВАННЯ КЛІЄНТІВ КОМПАНІЙ ІТ

1.1 Опис предметної області

У нашому світі є багато різних технічних та програмних продуктів, під час роботи з якими користувачі потребують допомоги у використанні та обслуговуванні, адже кожна людина має певні знання різного рівня та сфери, тому не завжди може самостійно розібратися в тому, як працює певний пристрій чи програма. Саме тому сфера обслуговування клієнтів ІТ-компаній є важливою, оскільки клієнтам необхідна служба підтримки для надання їм допомоги щодо використання технічних продуктів.

Технічний продукт — це будь-яке програмне чи апаратне забезпечення, яким користується клієнт, що потребує налаштування, а також, час від часу, обслуговування або ремонту. Прикладом технічного продукту є програмне забезпечення, сервери, комп'ютерна техніка та інші пристрої.

Служба підтримки продуктів ІТ — підрозділ або команда ІТ-компанії, що займається налаштуванням, обслуговуванням, ремонтом та діагностикою технічних продуктів для клієнтів. Підтримка може надаватися дистанційно (через аудіо- чи відеозв'язок) або під час виїзду працівника служби підтримки до клієнта [1].

Є дві основні ролі: працівник служби підтримки та клієнт.

- Клієнт — фізична або юридична особа, якій необхідна допомога з вирішенням її технічного питання. Він виконує наступні дії: подає заявку на візит працівника підтримки, зазначає дату та час, адресу, описує проблему, а після виконання робіт — отримує акт та здійснює оплату.
- Працівник служби підтримки — спеціаліст своєї сфери, який виконує технічні візити обслуговування продуктів компаній ІТ,

оновлює статуси заявок та призначає себе на них, заповнює акти виконаних робіт і веде звітність про виконані дії.

Існують наступні етапи для процесу надання послуги обслуговування клієнтів:

1) Запис на візит клієнтом (через телефон, сайт або іншу систему), під час якого вносяться в систему (або записуються в журнал) його дані, опис проблеми.
2) Призначення візиту (до даних заявки обирається дата та час обслуговування, коли клієнт та працівник підтримки буде вільний).
3) Виконання робіт на місці (спеціаліст підтримки виїжджає до клієнта, усуває його технічну проблему).
4) Оформлення акта виконаних робіт (працівник вносить інформацію про виконану роботу та візит у два акти виконаних робіт, клієнт підписує їх та отримує один екземпляр).
5) Отримання оплати (клієнт передає гроші працівнику за виконану роботу з обслуговування та його візит).

Запис на візит — процес, під час якого клієнт подає заявку для отримання допомоги від працівника служби підтримки з використанням технічного продукту, його налаштуванням або ремонтом. Мета запису — організація візиту працівника до клієнта, під час якого буде усунено проблему.

Коли у клієнта виникають проблеми з технічним продуктом, він звертається до служби підтримки та подає заявку на обслуговування, яка містить наступні його дані:

- прізвище, ім’я, по батькові;
- адресу візиту;
- номер телефону;
- короткий опис проблеми;
- дату і час для приїзду працівника підтримки.

Працівник служби підтримки вносить дані заявки клієнта до журналу візитів, що містить в собі інформацію про усі заявки на технічне обслуговування.

У вказаний час працівник приходить до клієнта за його адресою, усуває проблему або виконує інші дії для справної роботи технічного продукту. Потім він разом з клієнтом оформляє та підписує акт виконаних робіт у двох екземплярах, один з яких отримує клієнт, а інший буде зберігатися у компанії. Після цього клієнт оплачує працівникові вартість виконаних робіт, змінюється статус заяви, адже вона уже проведена, та спеціаліст служби підтримки відправляється в офіс або на наступне місце візиту.

Отже, обслуговування клієнтів є тривалим та неефективним процесом, тому в нинішній час багато компаній переходять на використання автоматизованих систем управління взаємовідносинами з клієнтами (CRM) та обліку обслуговування, що автоматизують процеси обліку, запису, обслуговування та ведення звітності. Це допоможе покращити контроль за виконанням робіт, обслуговуванням і комунікацією між працівниками та клієнтами. Організація обліку виїздів спеціалістів до клієнтів є одним з найважливіших завдань, а також важливою є реєстрація заявок, відстеження їхнього статусу, облік виконаних робіт та складання відповідної звітності.

1.2 Моделювання предметної області (UML)

У результаті аналізу предметної області визначено ключові об’єкти (клієнти, заявки та візити обслуговування, працівники служби підтримки, акти виконаних робіт, звіти), які взаємодіють між собою у сфері обслуговування. Автоматизоване управління цими об'єктами є предметом дослідження і реалізації програмного продукту. Такий підхід забезпечує структурованість та контрольованість процесів обслуговування, зменшуючи вплив людського фактора та покращуючи загальний рівень сервісу.

Після аналізу предметної області був побудований ряд моделей, у вигляді UML діаграм.

1.2.1 Діаграма прецедентів

Діаграма прецедентів предметної області має наступних акторів, що описані в таблиці 1.1.

Таблиця 1.1

Опис акторів

Актор	**Короткий опис**
Клієнт	Особа, яка потребує допомоги технічного спеціаліста. Клієнт замовляє візит працівника служби підтримки, повідомляє особисті дані, обирає дату та час, оплачує виконану роботу.
Працівник служби підтримки	Особа, яка надає технічну підтримку, виїзний спеціаліст, що вирішує проблеми клієнтів з технікою, заповнює акти виконаних робіт, вносить дані візитів до журналу та змінює їх статуси.

Всі цикли обслуговування, від створення заявки до оплати виконаних робіт, зображені на діаграмі прецедентів у додатку А.

1.2.2 Діаграма послідовності. На рис. 1 зображено діаграму послідовності предметної області, на якій зображено об’єкти системи в часі та їх взаємодію [2].

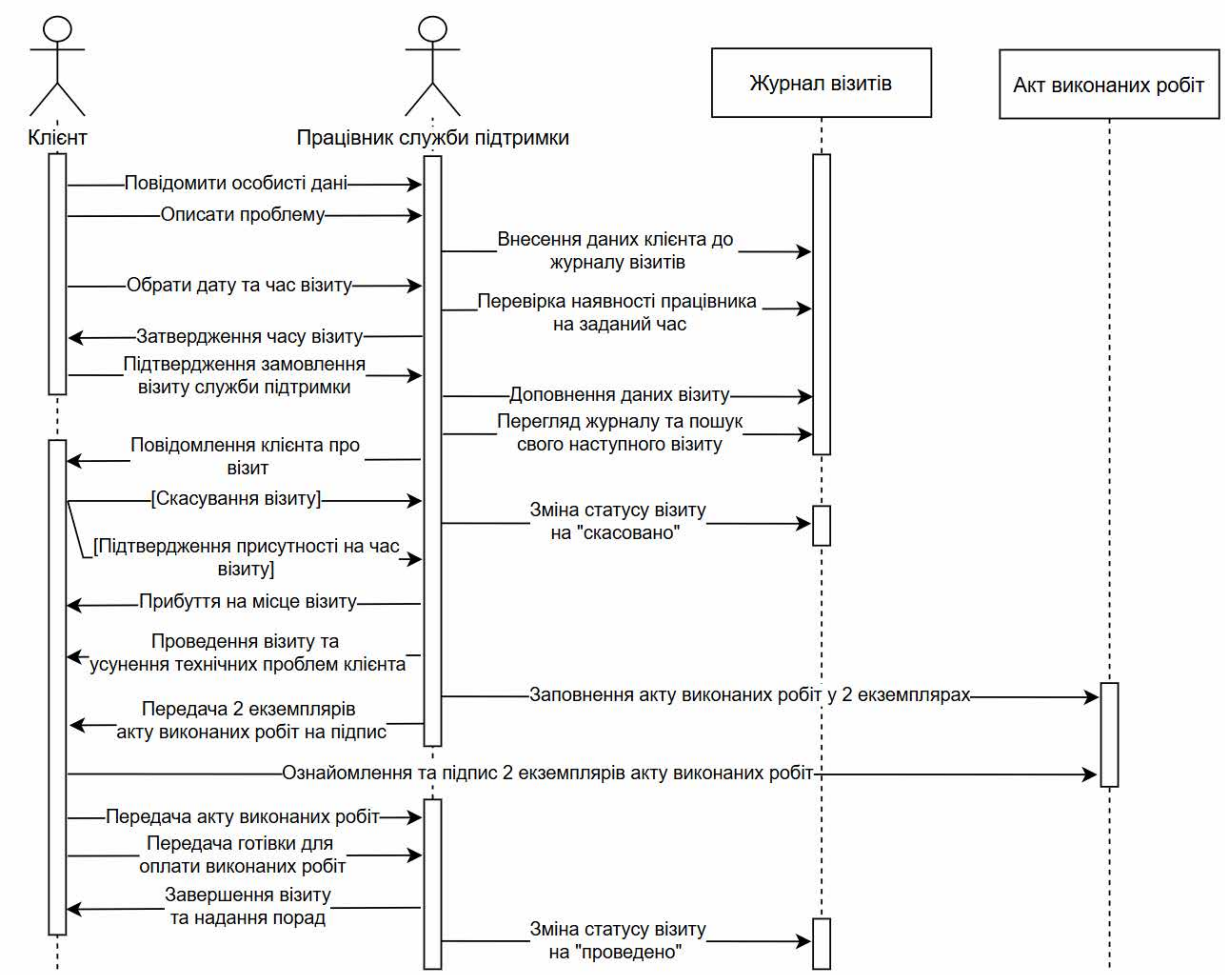


Рис. 1 Діаграма послідовності предметної області

1.2.3 Діаграма діяльності

Діаграму діяльності (activity diagram) предметної області зображено у додатку Б. Вона відображає дії у вигляді активностей з переходами між ними [3].

На діаграмі діяльності зображено процес організації візиту для обслуговування клієнта з метою вирішення технічних проблем виїзним працівником служби підтримки. Процес розділений двома доріжками, відповідно до двох основних ролей (працівника служби підтримки та клієнта).

Основні процеси:

1) Замовлення візиту — клієнт надає контактні дані та описує проблему, після чого обирає дату та час візиту.

2) Підтвердження замовлення візиту — після вибору дати та часу відбувається перевірка наявності виїзного працівника на заданий час. У разі відсутності вносяться зміни до часу візиту.
3) Сповіщення клієнта — при наявності вільного працівника на обраний час клієнта повідомляють про успішне оформлення візиту або про скасування, якщо працівник недоступний.
4) Підтвердження присутності — перед прибуттям працівник телефонує клієнту, щоб той підтвердив свою присутність у вказаний час або скасував візит, якщо не зможе з’явитись.
5) Візит та виконання робіт — працівник підтримки приїжджає на місце, вказане клієнтом, виконує роботи для усунення технічних неполадок та оформляє акт виконаних робіт у двох примірниках.
6) Завершення візиту — клієнт ознайомлюється з актом виконаних робіт, оплачує вказану в ньому вартість та підписує його, потім передає цей документ працівнику та отримує свій екземпляр. Потім працівник завершує візит та змінює його статус на «проведено».

Отже, на діаграмі діяльності було зображено взаємодію між клієнтом і працівником служби підтримки під час усіх етапів організації та проведення візиту обслуговування.

1.3 Огляд інформаційних джерел та існуючих рішень

Нині існує безліч різних CRM систем для управління клієнтськими відносинами та інших програмних застосунків які виконують подібні завдання, як запис на візит до психолога, бронювання місця в кінотеатрі, назначення візиту у лікаря, замовлення різних активностей та послуг чи призначення зустрічей, які можуть бути використаними для аналізу як зразок для розробки подібної ІС. Деякі популярні CRM (Customer Relationship Management) системи мають функціональність для обліку зустрічей, збереження інформації про клієнтів та історії взаємодій.

Огляд існуючих рішень:

1) Simplybook.me — безкоштовна система для бронювання послуг (рис. 2).

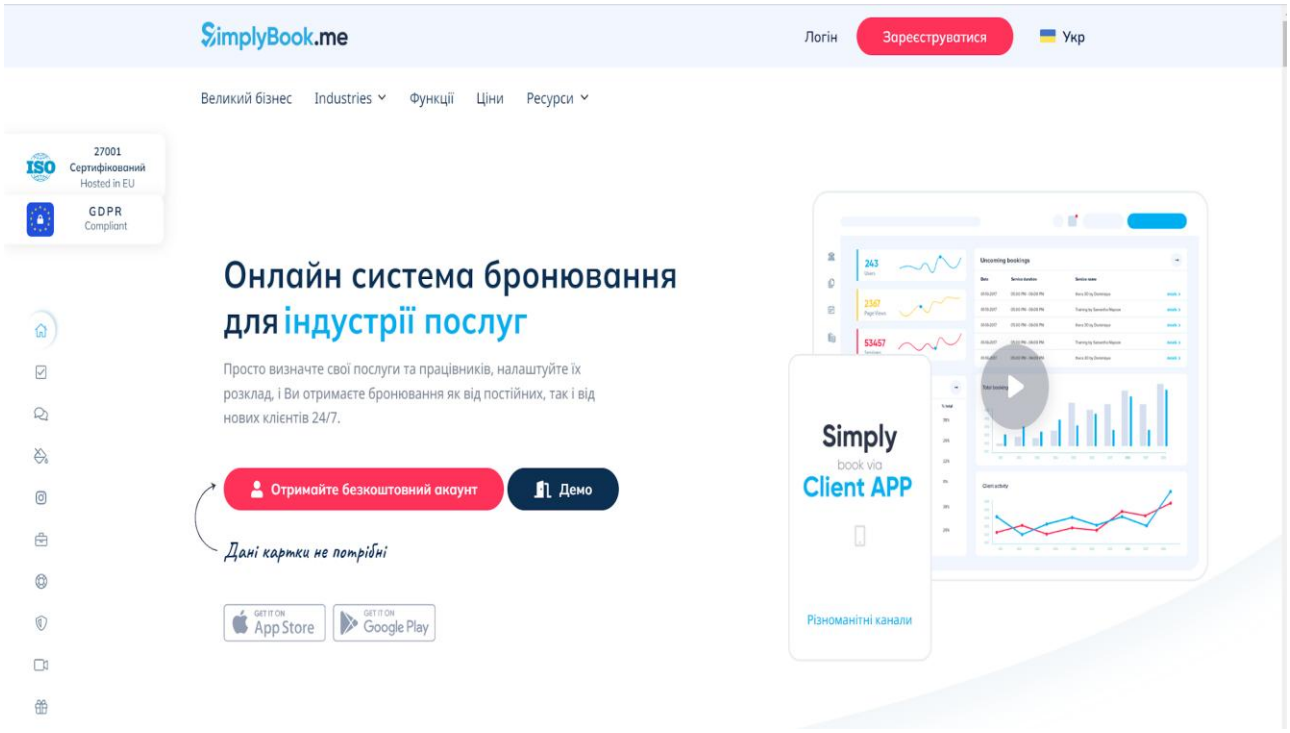


Рис. 2 Система бронювання Simplybook.me

Функціонал: онлайн бронювання; сповіщення через SMS, пошту; мобільний додаток клієнта та адміністратора; одержання платежів; інтеграція та API; наявність промокодів та сертифікатів; бізнес аналітика.

Переваги: безпека (є такі функції як HIPAA, SOAP, додатковий захист даних), щоденне резервне копіювання всіх даних, наявність фірмового застосунку для клієнта та адміністратора.

Недоліки: для використання повного функціоналу необхідна платна підписка, пробна версія містить не всі додаткові функції, має 50 бронювань та діє лише 14 днів [4].

2) EasyWeek — легка у використанні CRM система по управлінню бізнесом, що має спеціальний віджет та особисту веб-сторінку для кожного користувача, що працює цілодобово, з метою здійснення бронювань (рис. 3).

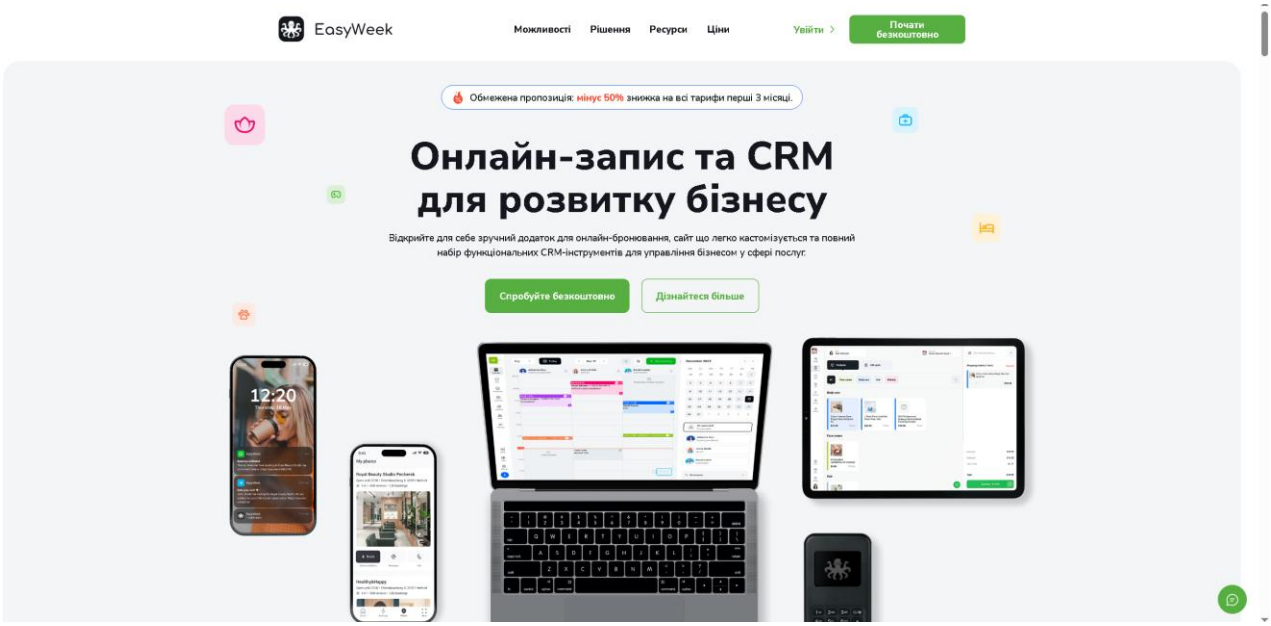


Рис. 3 Система автоматизації онлайн-запису EasyWeek

Функціонал: онлайн запис на прийом через веб-сторінку, електронний календар для планування візитів, облік товарів, статистика та аналітика, сповіщення про назначення запису, наявність зручних мобільних додатків з Push повідомленнями.

Переваги: швидкий пошук інформації про клієнта, статистики візитів і покупок, програми лояльності, формул бажаних послуг; індивідуальна онлайн веб-сторінка для кожного користувача; аналітика для оптимізації роботи.

Недоліки: складність інтеграції з багатьма іншими сервісами, для використання системи необхідна платна підписка [5].

1.4 Постановка завдання

Необхідно створити програмний застосунок для автоматизації процесів обслуговування клієнтів. Інформаційна система обліку обслуговування клієнтів компаній IT буде зберігати та обробляти дані візитів, заявок на обслуговування клієнтів, проведених робіт.

Функціонал інформаційної системи:

- подача заявок на візит обслуговування, що містять дату, час, місце, опис проблеми та тип ремонту, ID клієнта, а також можливість їх скасування для клієнтів;
- збереження даних обслуговування кожного клієнта;
- відображення заявок на обслуговування для працівників, можливість зміни їх статусу;
- облік актів виконаних робіт;
- автоматизація процесу генерації звітів та формування журналу заявок на візит.

1.4.1 Опис вимог

Функціональні:

1) Авторизація користувачів (за номером телефону та паролем).
2) Редагування контактних даних клієнта (електронна пошта, адреса).
3) Розподіл прав доступу відповідно до ролей (працівник, клієнт).
4) Подача клієнтом заявки на візит обслуговування (заявка містить дату та час, адресу, тип ремонту, опис проблеми, ID клієнта, ID заявки).
5) Скасування заявки на візит клієнтом у разі неможливості відвідування.
6) Підтвердження успішного додавання заявки на візит.
7) Запобігання дублюванню записів на основі унікальної комбінації ID клієнта та часу.
8) Перегляд працівниками заявок на візит, що мають статус «призначено» або «в процесі» (ID заявки, дата та час, адреса, номер телефону і ПІБ клієнта, тип ремонту, статус, опис проблеми).
9) Автоматичне оновлення розкладу після змін.
10) Встановлення статусу заявки на візит працівником (призначено, в процесі, скасовано, проведено).

11) Заповнення інформації про проведені візити працівником (на кожну заявку на зустріч клієнта назначається візит, що містить ID заявки, ID працівника, опис виконаної під час візиту роботи, дату та час, ID візиту). Якщо протягом одного візиту не було виправлено усі технічні проблеми, описані в заявці, то буде виконано ще один візит працівника служби підтримки. Після виправлення технічних проблем клієнта заповнюється акт виконаних робіт.
12) Заповнення акта виконаних робіт (працівник вносить наступні дані в систему: ID заявки на візит, дату, ID працівника, опис виконаних робіт, їх загальну вартість).
13) Генерація звітів для клієнтів, які міститимуть кількість їхніх проведених, скасованих заявок на візит за нинішній місяць.
14) Забезпечити функціональність генерації звітів для працівників технічної підтримки компаній IT:
    - із зазначенням кількості складених актів виконаних робіт кожним працівником та загальної суми виконаних робіт;
    - із кількістю оформлених заявок на візити клієнтом та загальною сумою здійснених оплат.
15) Експорт звітів у форматі .xlsx за допомогою спеціальної кнопки на інтерфейсі.
16) Підтвердження успішного внесення записів.

Нефункціональні:

1) Зрозумілий та зручний інтерфейс.
2) Захист персональних даних.
3) Швидкий відгук системи (до 3 секунд).

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних — абстрактне представлення структури бази даних [6]. Вона описує сутності з їх атрибутами, демонструє зв’язки між ними, не враховуючи фізичні особливості зберігання даних. Її створюють на основі вимог до інформаційної системи та використовують як основу для створення фізичної моделі БД [7].

Для побудови логічної моделі даних було створено ER (Entity-Relationship) діаграму, яка зображена на рис. 4. Вона показує основні сутності системи, їх атрибути та зв’язки [8], візуально демонструючи структуру і логіку інформаційної системи обліку обслуговування клієнтів компаній IT.

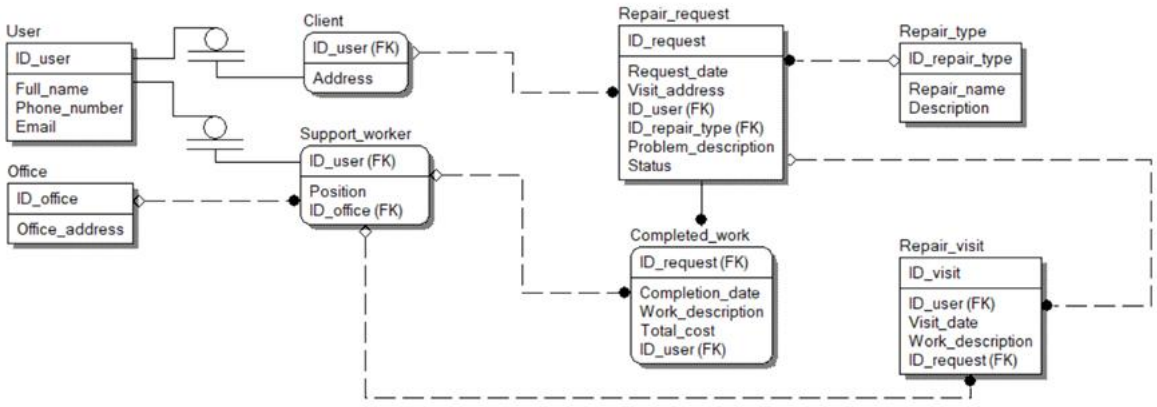


Рис. 4 ER діаграма

Опис:

- Сутність **Office** є незалежною, містить інформацію про адресу офісу. Атрибути: код офісу (ID_office) та його адреса (Office_address).
- Сутність **User** є незалежною, містить інформацію про клієнта та працівника служби підтримки та наслідується ними, має наступні атрибути: код користувача (ID_user), ПІБ (Full_name), номер мобільного телефону (Phone_number), електронну пошту (Email).

Client та **Support_worker** пов’язані з сутністю **User** ієрархією наслідування.

- Сутність **Client** є дочірньою, залежною, успадковує атрибути від таблиці **User** через ID_user та додатково містить атрибут адреси клієнта (Address).
- Сутність **Support_worker** є дочірньою, залежною, успадковує атрибути від таблиці **User** через код користувача (ID_user) та містить додатково атрибути посади (Position) та коду офіса (ID_office, FK від сутності **Office**).
- Сутність **Repair_type** є незалежною, містить типи/області проблем, з якими клієнту потрібна допомога. Має наступні атрибути: код типу (ID_repair_type), його назва (Repair_name) та опис (Description).
- Сутність **Repair_request** є залежною, містить інформацію про заявки на ремонт клієнтів. Має наступні атрибути: код заявки (ID_ request), ID_user (FK від **Client**), дату (Request_date), адресу (Visit_address), статус (Status), опис проблеми для усунення її під час візиту (Problem_description), код типу проблеми (ID_repair_type, FK від сутності **Repair_request**). Ця таблиця містить зовнішні ключі для зв’язку з таблицями **Client** та **Repair_type**.
- Сутність **Completed_work** є залежною, містить інформацію про виконану роботу. Має наступні атрибути: код заяви (ID_request, FK від **Repair_request**), дата виконання роботи та закриття заявки (Completion_date), опис виконаних робіт (Work_description), сума для оплати за візити обслуговування по заявці клієнта (Total_cost), код користувача (ID_user, FK від **Support_worker**). Ця таблиця містить зовнішні ключі для зв'язку з таблицями **Support_worker** та **Repair_request**.
- Сутність **Repair_visit** є залежною, містить інформацію про ремонти та консультації для надання підтримки клієнтам. Має наступні

атрибути: код візиту (ID_visit), код користувача (ID_user, FK від **Support_worker**), дату візиту (Visit_date), опис виконаної роботи під час візиту (Work_description), код заявки (ID_request, FK від **Repair_request**). Ця таблиця містить зовнішні ключі для зв'язку з таблицями **Support_worker** та **Repair_request**.

Нормалізація. Ця діаграма є нормалізованою до третьої нормальної форми (3NF), адже атрибути є атомарними (містять неподільні значення), відсутні повторювані групи або масиви значень, присутня повна залежність неключових атрибутів від первинного ключа, немає як часткових залежностей, так і транзитивних [9].

2.2 Вибір системи управління інформаційною базою

Відповідно логічній моделі, при виборі СУБД необхідно зупинитися на реляційній структурі.

Система управління базами даних (СУБД) — програмне забезпечення, за допомогою якого створюють, зберігають, змінюють та використовують БД, а також виконують управління цілісністю, безпекою та доступом до даних [10].

Реляційна БД складається з таблиць, кожна з яких представляє деяку сутність (наприклад, працівника служби підтримки, ремонтний візит), між ними встановлюються зв’язки за допомогою первинних та зовнішніх ключів [11].

Сьогодні популярними є наступні реляційні СУБД:

1) Microsoft Access. MS Access — це СУБД від Microsoft, створена для локального використання.

Його функціональні можливості:

- інтеграція з продуктами Microsoft Office;
- автоматизація завдань за допомогою макросів [12];
- швидке створення простих баз даних, звітів, форм та запитів;
- розробка прототипів систем.

Переваги:

- можуть використовувати люди без знань SQL [13];
- швидкий запуск проекту;
- зручний графічний інтерфейс.

Недоліки:

- немає підтримки повноцінної клієнт-серверної модель;
- наявність обмеження кількості одночасних підключень;
- погана масштабованість для великих обсягів даних;
- відсутність підтримки зовнішніх додатків на C# та інших мовах програмування.

Отже, MS Access не підходить для мого проекту, адже він має централізоване зберігання даних, необхідність доступу з клієнтського застосунку.

2) Oracle. Oracle — об’єктно-реляційна та комерційна СУБД, яка є найпродуктивнішою, орієнтованою на великі корпоративні рішення. Ця система управління БД підтримує реляційну модель, масштабування, розподілену обробку даних, має гнучкі механізми безпеки [14].

Його перевагами є:

- висока продуктивність;
- розширена функціональність адміністрування та резервного копіювання;
- підтримка складних транзакцій [15].

Недоліки:

- складність у налаштуванні;
- висока ціна ліцензії;
- надлишкова функціональність для невеликих та середніх проєктів.

У моєму випадку Oracle не є оптимальним вибором СУБД через його складність обслуговування, зайві функціональні можливості і необхідність

оплати, тому замість нього обрано безкоштовний, легкий у налаштуванні та ефективний Microsoft SQL Server.

3) Microsoft SQL Server. Microsoft SQL Server — потужна реляційна СУБД, яка підтримує клієнт-серверну модель, що дозволяє ефективно обробляти великі обсяги даних.

Переваги:

- масштабованість — підходить як для великих, так і малих проєктів;
- має дружній для користувачів інтерфейс;
- підтримує інтеграцію з продуктами Microsoft;
- забезпечує оптимізацію складних запитів та цілісність зв’язків;
- швидко обробляє великі обсяги даних та має механізми для контролю доступу до даних за допомогою різних прав та ролей, системних та створених адміністратором;
- має безкоштовну версію Microsoft SQL Server Express.

Недоліки:

- складність адміністрування;
- обмеження розміру бази даних та функціоналу у безкоштовній версії Express [16].

Через потребу у розміщенні бази даних на віддаленому сервері для взаємодії з системою було обрано клієнт-серверну архітектуру [17]. Вона використовується для:

- забезпечення централізованого доступу до даних;
- організації роботи з БД для багатьох користувачів з різних місць через мережу «Інтернет»;
- безпеки та контролю доступу;
- гнучкого масштабування системи.

Для реалізації інформаційної системи обліку обслуговування клієнтів компаній ІТ, було обрано СУБД Microsoft SQL Server, його використання спростить розробку клієнт-серверного застосунку через простоту інтеграції з

технологіями .NET та C#. Таким чином, SQL Server є оптимальним вибором для створення цієї ІС, забезпечуючи зручність, швидкість, стабільність та безпеку роботи з базою даних, що показано на рис. 5.

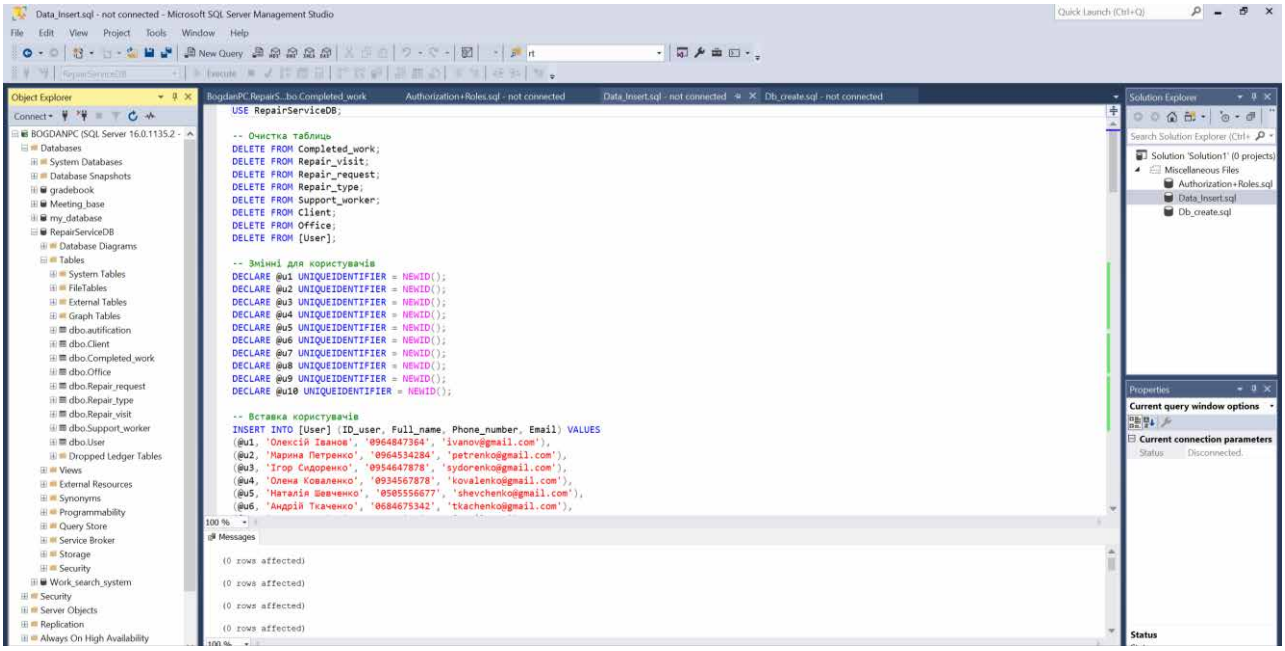


Рис. 5 Інтерфейс Microsoft SQL Server Management Studio

Для забезпечення клієнт-серверної архітектури БД була розміщена у хмарі «Google Cloud», завдяки чому забезпечено віддалений доступ до бази даних із будь-якого місця, яке має доступ до інтернету. Це дуже зручно для розробки та роботи системи, а також для її масштабованості.

2.2.1 Пояснення визначень. Хмара (cloud) — віддалений сервер або набір серверів, що надають ресурси для обчислення та зберігання через Інтернет [18].

Bucket — контейнер для зберігання об’єктів (файлів), що використовується у Google Cloud Storage.

Екземпляр SQL Server — окремий запущений сервер, який працює незалежно від інших екземплярів на тому ж сервері та має власні налаштування, бази даних, ресурси і обробляє запити клієнтів.

Порт 1433 — порт для підключення до Microsoft SQL Server.

IP-адреса — унікальний числовий ідентифікатор, що використовують як адресу пристрою в мережі.

2.2.2 Створення хмарної архітектури. Нижче перерахована послідовність дій створенні хмарної архітектури.

1) Створення екземпляра сервера баз даних (SQL Server Instance). Було створено окремий SQL Server у хмарному середовищі. Він є окремою одиницею обчислювальної потужності, що виконує роль віддаленого сервера баз даних. Його функціонал: збереження, обробка та обслуговування запитів до БД у режимі реального часу. Створений екземпляр було названо «repairservicedbinstance», його типом СУБД є Microsoft SQL Server.
2) Створення сховища (bucket). Для екземпляра сервера баз даних було створено bucket — хмарне сховище в Google Cloud Storage, що створене для збереження файлів БД, всередині якого зберігають файл бази даних у форматі .bak або .mdf/.ldf. Завдяки цьому хмарному середовищу локальну базу даних можна легко та швидко перенести на хмарний сервер.
3) Завантаження файлу бази даних у bucket. Файл БД у форматі .bak було завантажено у створене сховище, а потім bucket приєднано до екземпляра сервера баз даних за допомогою інструментів Google Cloud.
4) Налаштування доступу до сервера бази даних. Для забезпечення публічного доступу до віддаленої бази даних відкрито мережевий порт (TCP 1433), по якому SQL Server приймає запити. Обрано IP-адресу 130.211.53.34, яка є зовнішньою адресою екземпляра сервера баз даних, а також надано можливість публічного доступу до серверу БД.

На рис. 6 показано вкладку встановлення зв’язку у SQL Google Cloud.

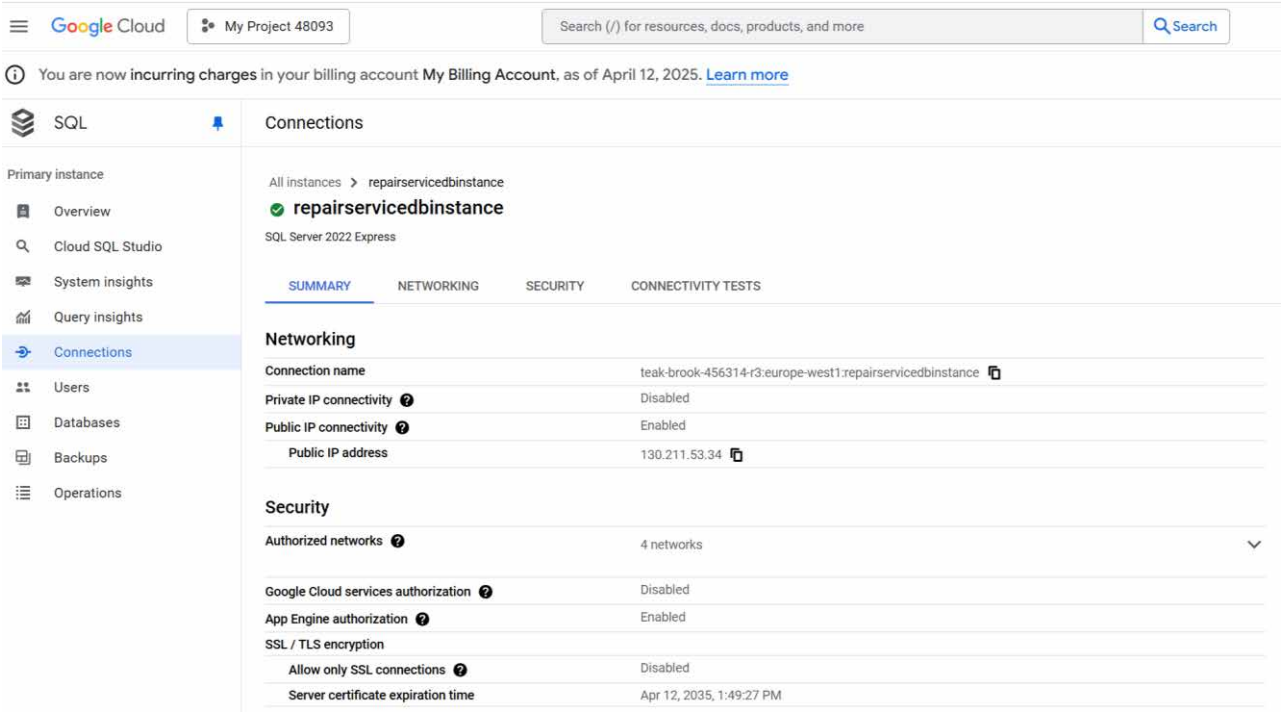


Рис. 6 Встановлення зв’язку у SQL Google Cloud

2.3 Створення бази даних

Було створено реляційну базу даних «RepairServiceDB» в середовищі Microsoft SQL Server для реалізації інформаційної системи обліку обслуговування клієнтів компаній IT.

2.3.1 Створення бази даних та її таблиць. Відповідно до ER діаграми (див. рис. 4), було створено БД та її таблиці. На рис. 7 показано фрагмент коду, що створює базу даних та її таблицю. Повний код створення таблиць бази даних знаходиться у додатку В на сторінках 1-3.

Були створені такі таблиці:

- User, що зберігає загальні дані користувачів;
- Client, що містить дані клієнтів (адресу);
- Support_worker, який зберігає дані працівників служби підтримки (посаду та код офісу);
- Office — містить дані про офіси компанії служби підтримки;

- Repair_type, в якому зберігаються типи існуючих ремонтів;
- Repair_request, що зберігає дані клієнтів для проведення візиту обслуговування;
- Repair_visit, який має дані візиту обслуговування;
- Completed_work, в якій збережена інформація по візитах клієнта і виконаній на них роботі працівником по заявці клієнта на ремонт.

Db_create.sql - BOG...erviceDB2 (sa (56))

```
CREATE DATABASE RepairServiceDB;
GO

USE RepairServiceDB;
GO

-- Видалення таблиць у правильному порядку
IF OBJECT_ID('Completed_work', 'U') IS NOT NULL DROP TABLE Completed_work;
IF OBJECT_ID('Repair_visit', 'U') IS NOT NULL DROP TABLE Repair_visit;
IF OBJECT_ID('Repair_request', 'U') IS NOT NULL DROP TABLE Repair_request;
IF OBJECT_ID('Repair_type', 'U') IS NOT NULL DROP TABLE Repair_type;
IF OBJECT_ID('Support_worker', 'U') IS NOT NULL DROP TABLE Support_worker;
IF OBJECT_ID('Client', 'U') IS NOT NULL DROP TABLE Client;
IF OBJECT_ID('Office', 'U') IS NOT NULL DROP TABLE Office;
IF OBJECT_ID('[User]', 'U') IS NOT NULL DROP TABLE [User];
GO

-- Таблиця користувачів
CREATE TABLE [User] (
    ID_user UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
    Full_name VARCHAR(100) NOT NULL,
    Phone_number VARCHAR(20),
    Email VARCHAR(MAX)
);
```

Рис. 7 Створення БД та її таблиці для користувачів

На рис. 8 зображено фізичну модель бази даних.

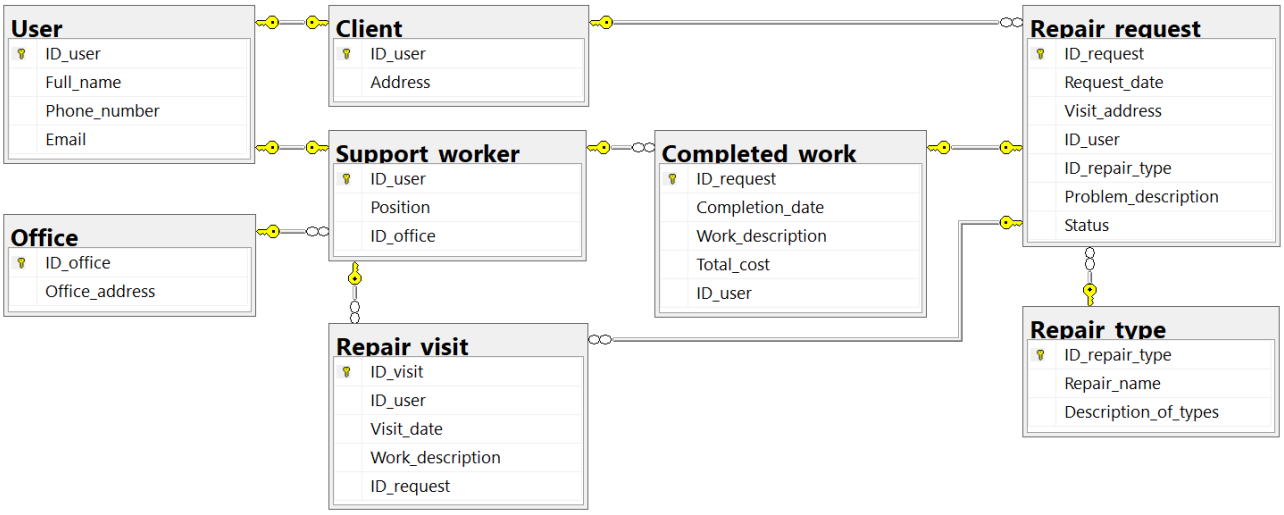


Рис. 8 Фізична модель даних

2.3.2 Процедури БД

Процедура — збережена послідовність SQL-команд, що має можливість багаторазового виклику та виконується на сервері [19].

Всього створено вісім процедур:

1) Процедура «AddClient» має чотири параметри, за допомогою яких додається клієнт до таблиць User та Client:
   - FullName — ПІБ клієнта;
   - Phone — номер телефону;
   - Email — електронна пошта;
   - Address — адреса проживання.
2) Процедура «AddWorker», за допомогою якої відбувається додавання нового працівника до таблиць User та Support_worker, має п’ять параметрів:
   - FullName — ПІБ працівника,
   - Phone — номер телефону,
   - Email — електронна пошта,
   - Position — посада,
   - OfficeID — ідентифікатор офісу.

3) Процедура «AddRepairRequest» додає нову заявку на ремонт в таблицю Repair_request та має шість параметрів:
   - ClientID — ідентифікатор клієнта,
   - RequestDate — дата призначення візиту,
   - VisitAddress — адреса візиту,
   - RepairTypeID — тип ремонту,
   - ProblemDescription — опис проблеми,
   - Status — статус заявки.
4) Процедура «ScheduleRepairVisit» записує інформацію про візит працівника служби підтримки у таблицю Repair_visit, має чотири параметри:
   - WorkerID — ідентифікатор працівника,
   - VisitDate — дата візиту,
   - WorkDescription — опис виконаної роботи під час візиту,
   - RequestID — ідентифікатор заявки.
5) Процедура «CompleteRepair», що записує завершення ремонту в таблицю Completed_work, має п’ять параметрів:
   - RequestID — ідентифікатор заявки,
   - WorkerID — ідентифікатор працівника,
   - CompletionDate — дата завершення виконання роботи по заявці клієнта,
   - WorkDescription — опис виконаної роботи,
   - TotalCost — загальна вартість ремонту.
6) Процедура «GetClientRepairReport» не має параметрів та формує звіт по клієнтах, що містить кількість завершених заявок ремонтів та сумарну оплату за них для кожного клієнта окремо.
7) Процедура «GetWorkerPerformanceReport» не має параметрів. Вона створює звіт по всіх завершених ремонтах по кожному з працівників

служби підтримки, вказуючи кількість закритих ним заявок та отриману вартість за їх виконання.

8) Процедура «GetClientMonthlyVisitStats», яка рахує кількість проведених робіт по заявкам клієнта, а також скасованих за нинішній місяць, має 3 параметри:
   - ClientID — ідентифікатор клієнта;
   - Year — рік;
   - Month — місяць.

Коди створення процедур представлені в додатку В на сторінках 3-7. На рис. 9 представлено приклад створення процедури для додавання клієнта.

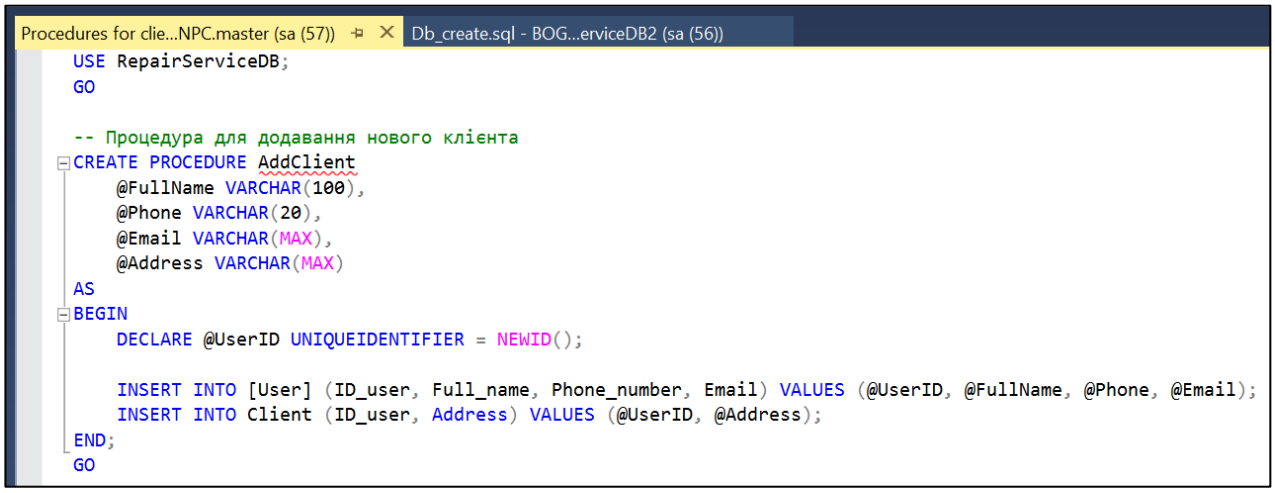


Рис. 9 Процедура для додавання клієнта

2.3.3 Тригери. Тригер — особлива збережена процедура, що автоматично виконується при виникненні певної події (INSERT, UPDATE, DELETE) у базі даних [20]. Всього створено чотири тригера.

1) Тригер «trg_CheckWorkerOffice».

   Подія — AFTER INSERT, UPDATE на таблиці Support_worker.

   Призначення — перевірити існування вказаного офісу у таблиці Office перед додаванням або оновленням даних працівника.

   Дія — якщо офіс не існує, виводиться повідомлення про помилку.

Ціль — забезпечити коректність зв’язків між працівниками та офісами та цілісність даних.

2) Тригер «trg_DeleteUserOnClientDelete».

Подія — AFTER DELETE на таблиці Client.

Призначення — автоматичне видалення інформації про клієнта з таблиці User після виконання аналогічної дії в таблиці Client.

Дія — пошук ID клієнта в DELETED та видалення його даних з таблиці User.

Ціль — повне видалення особистих даних клієнтів з таблиць для уникнення «висячих» користувачів.

3) Тригер «trg_DeleteUserOnSupportWorkerDelete».

Подія — AFTER DELETE на таблиці Support_worker.

Призначення — при видаленні даних працівника служби підтримки з таблиці Support_worker виконати аналогічну дію в таблиці User.

Дія — аналогічно до попереднього тригера, шукає ID працівника в DELETED та видаляє його дані в таблиці User.

Ціль — видалення даних працівників з таблиці користувачів при їх звільненні (щоб не зберігати зайву інформацію).

4) Тригер «PreventRepairRequestDeletion».

Подія — INSTEAD OF DELETE на таблиці Repair_request.

Призначення — заборона видалення заявок, які мають статус «проведено».

Дія — якщо хтось намагається видалити заявку з статусом «проведено», то відбувається скасування запиту та виводиться помилка.

Ціль — збереження даних про проведені ремонти та консультації по заявкам клієнтів.

Коди створення тригерів наведено в додатку В на сторінках 7-9.

На рис. 10 показано код тригера, що перевіряє існування вказаного офісу перед додаванням працівника.

```
USE RepairServiceDB;
GO

-- Дроп тригера перед створенням
IF OBJECT_ID('trg_CheckWorkerOffice', 'TR') IS NOT NULL
    DROP TRIGGER trg_CheckWorkerOffice;
GO

-- Створення тригера для перевірки офісу працівника
CREATE TRIGGER trg_CheckWorkerOffice
ON Support_worker
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @OfficeID UNIQUEIDENTIFIER;

    -- Отримання ID офісу з таблиці після вставки чи оновлення
    SELECT @OfficeID = ID_office FROM INSERTED;

    -- Перевірка чи існує такий офіс
    IF NOT EXISTS (SELECT 1 FROM Office WHERE ID_office = @OfficeID)
    BEGIN
        PRINT 'Error: office not found for the worker.';
    END
END;
GO
```

Рис. 10 Тригер для перевірки існування офісу перед додаванням працівника

2.3.4 Створення ролей та користувачів

Role-Based Access Control (RBAC) — механізм управління доступом, що базується на ролях, за допомогою якого кожному користувачу надається необхідний функціонал відповідно до його ролі у системі. Таким чином, усі клієнти отримають можливість запису на обслуговування, а працівники — заповнення даних проведених візитів, обліку виконаних робіт та інших дій, необхідних для виконання своїх бізнес-задач. RBAC дозволяє захистити систему та надавати права доступу групам користувачів, а не кожному окремо. Його

використання дозволяє усунути можливість несанкціонованого доступу та покращити безпеку системи [21].

Основні ролі. Створено дві основні ролі та визначено їх права:

- ClientRole — для клієнтів сервісу;
- SupportWorkerRole — для працівників служби підтримки.

Можливості клієнта в системі:

- подача заявки на замовлення візиту;
- перегляд інформації про власні візити;
- скасування своїх заявок;
- формування звіту по своїм візитам;
- зміна особистих контактних даних (електронної пошти та адреси).

Клієнти не мають прав на перегляд чи зміну інформації інших користувачів та працівників, доступу до адміністрування. Це забезпечується надання їм прав на рівні бази даних на вибірку (SELECT), вставлення даних (INSERT) та оновлення (UPDATE), без права на видалення (DELETE), створення об’єктів БД (наприклад: користувачів, таблиць, процедур та інших).

Роль працівника служби підтримки має більший функціонал, адже вони обслуговують клієнтів та виконують більше процесів всередині системи. Вони мають наступний функціонал:

- перегляд заявок на обслуговування;
- редагування даних заявок, змінюючи їх статус;
- заповнення інформації проведених візитів;
- внесення інформації про виконані послуги;
- формування звітів по роботі працівників служби підтримки, витратам клієнтів.

Ця роль (SupportWorkerRole) має права «SELECT», «INSERT», «UPDATE», «DELETE» у БД.

На рис. 11 показано код створення ролей та призначення їм прав.

```
USE RepairServiceDB;
GO
-- Створення ролей
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'ClientRole')
   CREATE ROLE ClientRole;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name =
'SupportWorkerRole')
   CREATE ROLE SupportWorkerRole;
GO
-- Призначення прав ролям
GRANT SELECT, INSERT, UPDATE ON SCHEMA :: dbo TO ClientRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA :: dbo TO
SupportWorkerRole;
```

Рис. 11 Код створення ролей та призначення прав

Створення користувачів. Для створення користувачів системи та їх авторизації у застосунку «ІС обліку обслуговування клієнтів компаній ІТ» було сформовано особливу таблицю «autification», всередині якої знаходяться наступні дані:

- логін (номер телефону);
- пароль;
- роль користувача.

На основі записів у цій таблиці автоматично створюються користувачі у Microsoft SQL Server з відповідними до їх ролі правами.

За допомогою розподілу функціоналу і доступу до об’єктів БД відбувається спрощення адміністрування та запобігання несанкціонованого доступу іншими людьми, які не мають прав на певні дії. Таким чином забезпечується високий рівень безпеки та контролю доступу системи, адже кожен користувач має свій логін та пароль і чітко визначені права доступу відповідно його ролі.

Повний код створення ролей та прав доступу, користувачів БД представлений у додатку В на сторінках 9-11.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

За допомогою діаграми пакетів на рис. 12 зображено організаційну структуру програмного забезпечення.

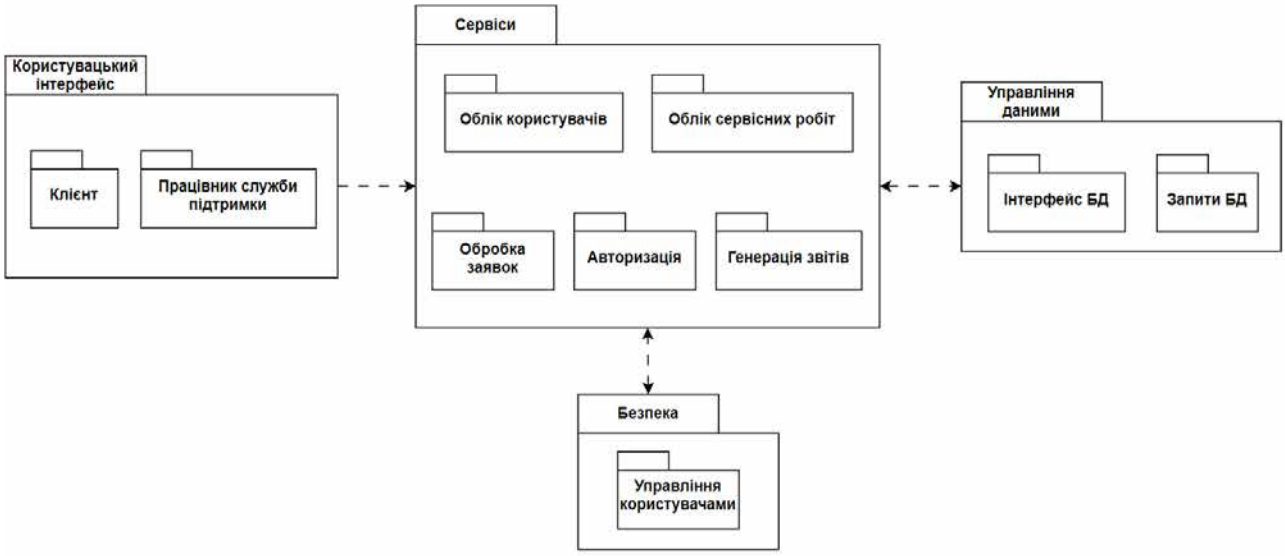


Рис. 12 Діаграма пакетів

Опис пакетів

1) **Користувацький інтерфейс (UI)**. Цей пакет реалізує взаємодію кінцевого користувача з програмою. Він розділений на два підпакети:

- **Клієнт**, що містить візуальні елементи, які будуть використані для реалізації функціоналу подання заявки, їх перегляду, редагування особистих контактних даних, генерації звіту по своїм проведеним та скасованим візитам;
- **Працівник служби підтримки**, що відображає взаємодію БД з back-end частиною проекту для обробки заявок, формування звітів по працівникам та клієнтам, зміни статусу заявок, внесенню даних по виконаній роботі.

2) **Сервіси (Business Logic).** Пакет, всередині якого реалізовано основну бізнес-логіку програми.

Складається з таких підпакетів:

- **Облік користувачів**, за допомогою якого відбувається збереження та обробка інформації про клієнтів і працівників;
- **Облік сервісних робіт**, який зберігає дані про звернення клієнтів та інформацію про виконану роботу;
- **Обробка заявок**, що містить реалізацію прийому, обробки, оновлення та завершення заявок на обслуговування, які подають клієнти;
- **Авторизація**, функціоналом якої є визначення ролі користувача та надання йому доступу до інтерфейсу, відповідного до його прав у системі;
- **Генерація звітів**, яка виконує формування звітів про статистичні дані візитів для працівників та клієнтів, використовуючи запити до БД.

3) **Безпека (Security)**. Вона відповідає за контроль доступу до функціоналу системи:

- **Управління користувачами**, перевіряючи права доступу користувачів на основі введеного логіну та пароля, гарантуючи надійність і точність надання необхідного функціоналу клієнтам та працівникам відповідно до їх даних авторизації.

4) **Управління даними (Data Layer) —** пакет за допомогою якого відбувається реалізація роботи з базою даних.

Управління даними складається з наступних підпакетів:

- **Інтерфейс БД**, що використовується для встановлення з’єднання з базою даних, відкриття та закриття її підключень;
- **Запити БД**, всередині якого зберігаються SQL-запити для додавання, отримання, оновлення, видалення даних.

Взаємозв’язки між пакетами

- Пакет **Користувацький інтерфейс** взаємодіє з **Сервісами**, викликаючи функції та передаючи для них вхідні дані, потім отримуючи результати та відображаючи їх.
- **Сервіси** використовують **Управління даними** для отримання інформації з бази даних, її оновлення або видалення, запису у БД.
- Пакет **Безпека** взаємодіє з **Сервісами** для коректного функціонування авторизації користувачів і обмеження доступу до системи відповідно до ролей.

3.2 Вибір інструментарію для створення ППЗ

Під час створення прикладного програмного забезпечення було використано сучасний інструментарій, що забезпечує зручну реалізацію функціоналу, масштабованість системи, простоту її розгортання, адміністрування та надання доступу до даних за допомогою хмарного серверу даних.

Було обрано мову програмування C#, адже вона:

- підтримує об’єктно-орієнтований підхід;
- чудово працює з платформою .NET;
- має високу продуктивність.

Для побудови графічного інтерфейсу користувача використано платформу .NET Framework із Windows Forms, за допомогою яких було швидко та легко створено графічний інтерфейс, що містить кнопки, таблиці, текстові поля, заголовки та інші елементи управління.

Microsoft Visual Studio обрано у ролі середовища розробки програмного застосунку, адже він є популярним та потужним інструментом з зручним інтерфейсом, можливістю тестування та налагодження, а також інтегрується з Microsoft SQL Server.

За допомогою технології ADO.NET реалізована передача даних між програмою та базою даних, адже він забезпечує стабільний зв’язок, обробку запитів і оновлення інформації у базі.

За допомогою перелічених вище інструментів побудовано масштабовану систему з можливістю віддаленого доступу до даних завдяки хмарному серверу БД.

3.3 Реалізація інтерфейсної частини

Інтерфейсну частину було реалізовано за допомогою Windows Forms (WinForms) на мові програмування C#, створено форми для клієнта, працівника служби підтримки, авторизації. Метою інтерфейсу є забезпечення зручного та інтуїтивно зрозумілого доступу до функціоналу інформаційної системи обліку обслуговування клієнтів компаній IT.

Інтерфейс містить такі основні елементи керування:

- Кнопки (Button), що використовуються для виклику певної події/виконання дій. Наприклад:
  - кнопка з зображенням хрестика виконує закриття програми та з’єднання з БД, яка виконує дію при натисканні (виклик методу button13_Click);
  - кнопка авторизації, при натисненні якої перевіряється наявність користувача за введеним логіном та паролем та його роль, при наявності якої відкривається відповідна до неї форма (працівника або клієнта), інакше — виводиться повідомлення про помилку.
- Мітки (Label), призначенням яких є відображення текстової інформації. Наприклад:
  - головний заголовок форми: «Інформаційна система обліку обслуговування клієнтів компаній IT»;
  - контактна інформація служби підтримки;

- підписи до текстових полів та інші.

- Текстові поля (TextBox) для введення і виведення тексту. Наприклад:
  - для показу ідентифікатору клієнта/працівника (лише для читання) знизу інтерфейсу;
  - для введення загальної вартості виконаних робіт.
- Випадаючі списки (ComboBox), за допомогою яких користувач обрає один із перелічених варіантів. Їх було використано для вибору:
  - коду заявки клієнта;
  - виду робіт;
  - статусу.
- Багаторядкові поля (RichTextBox), які створені для введення та відображення розширеного тексту. Наприклад:
  - richTextBox1 для показу адреси;
  - richTextBox2 призначене для опису виконаних робіт.
- Поле вибору дати та часу (DateTimePicker) — спеціальний елемент керування з випадаючим календарем для зручного вибору дати та часу. Його було використано для вибору дат в:
  - заявках на візит;
  - обліку проведених візитів;
  - актах виконаних робіт.
- Блоки для групування елементів (GroupBox) — елемент що об’єднує пов’язані елементи. Наприклад, всередині groupBox2 знаходяться елементи які стосуються акта виконаних робіт.
- Вкладка (TabPage) — дочірній елемент керування TabControl, що використовується для розділення функціоналу на логічні блоки (вкладки). Вкладками TabControl1 для клієнта є:
  - контактні дані;
  - візит;
  - звіт.

Інтерфейс було побудовано відповідно до вимог, зручності, інтуїтивності керування та доступності: логічно розташовано елементи, використано зрозумілі шрифти, випадаючі списки для уникнення помилок введення та автозаповнення деяких текстових полей інформацією (ID_user, Address), а також підказки при наведенні для користувачів. Таблиці журналу візитів у працівників та клієнтів мають повзунок та можливість зміни розміру при розтягненні границь колонок та стовбців, при виборі комірки можна виділити та копіювати дані з неї.

На рис. 13 показано приклад елементів інтерфейсу.

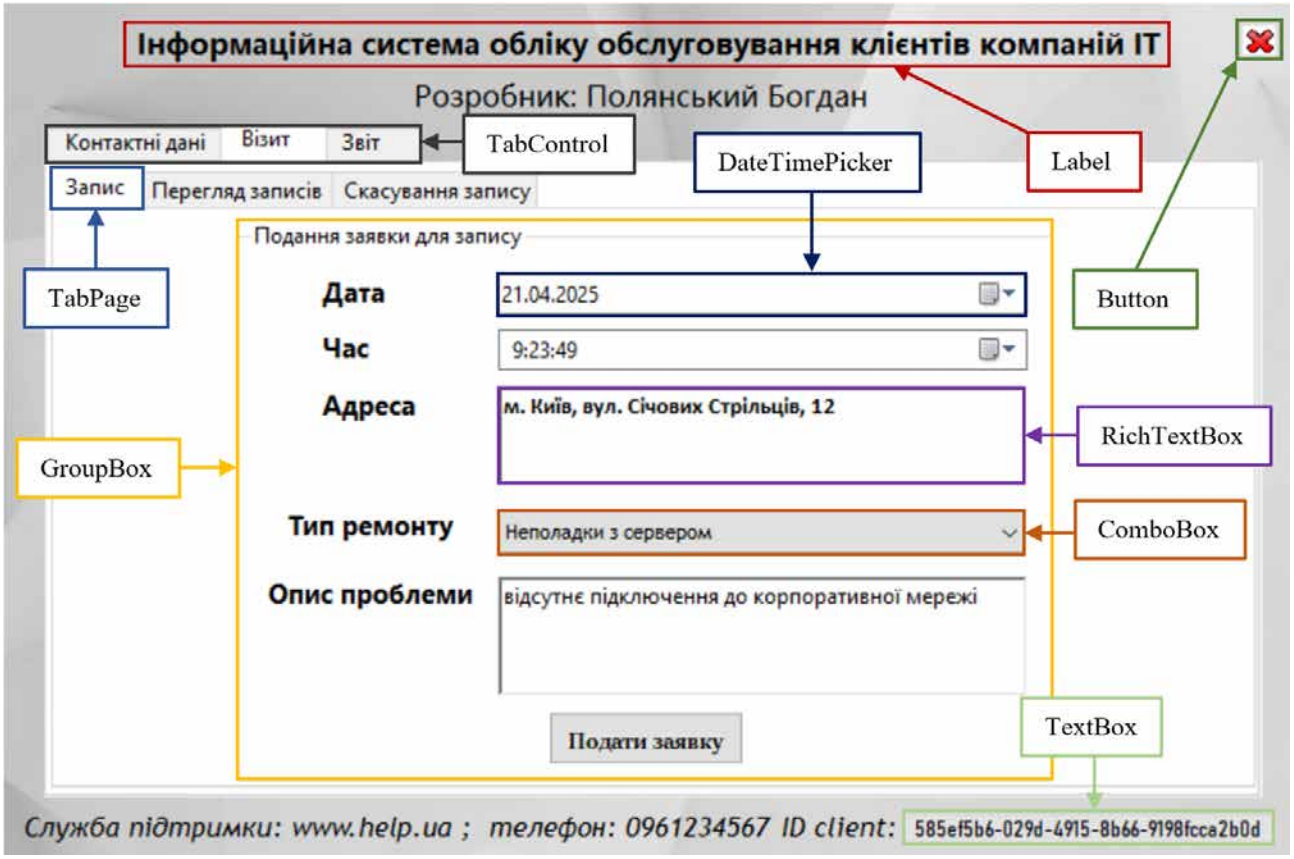


Рис. 13 Приклад елементів інтерфейсу

3.4 Проектування і реалізація сервісів

Сервіси — функціональні частини програми, які містять основні процеси системи обліку обслуговування клієнтів та реалізують їх функціональні можливості, взаємодіючи з графічним інтерфейсом для отримання даних та їх відображення. Вони представлена у вигляді сервісних класів та функцій,

інтегрованих у відповідні форми: Login_form.cs, Client_form.cs, Worker_form.cs, кожна з яких представляє функціонал відповідно до її бізнес-завдання та виконує дії з базою даних за допомогою підключення SqlConnection, запитів та процедур.

Перелік сервісів у програмі:

1) **Авторизація** (форма **Login_form.cs)**. Її основна ціль — визначити роль особи, що ввела номер телефону та пароль, а потім відкрити відповідний інтерфейс.

   Основні функції:

   - **Перевірка ролі користувача** (метод **GetRole**). Відбувається перевірка наявності користувача за введеним логіном та паролем в базі даних. Якщо його знайдено — повертається відповідна роль («SupportWorkerRole» або «ClientRole») для майбутнього відкриття відповідної інтерфейсної форми, інакше — виведення системою повідомлення про помилку через неправильність введених даних авторизації. Код методу перевірки ролі зображено на рис. 14.

```
public static string GetRole(SqlConnection connection, string phoneNumber)
        {  string queryClient = @"
              SELECT 'ClientRole'
              FROM Client C
              JOIN [User] U ON U.ID_user = C.ID_user
              WHERE U.Phone_number = @PhoneNumber";
           string querySupportWorker = @"
              SELECT 'SupportWorkerRole'
              FROM Support_worker SW
              JOIN [User] U ON U.ID_user = SW.ID_user
              WHERE U.Phone_number = @PhoneNumber";
           using (SqlCommand cmd = new SqlCommand(querySupportWorker, connection))
           {  cmd.Parameters.AddWithValue("@PhoneNumber", phoneNumber);
              var result = cmd.ExecuteScalar();
              if (result != null)
                 return result.ToString(); }
           using (SqlCommand cmd = new SqlCommand(queryClient, connection))
           {  cmd.Parameters.AddWithValue("@PhoneNumber", phoneNumber);
              var result = cmd.ExecuteScalar();
              if (result != null)
                 return result.ToString(); }
           return string.Empty;
        }
```

Рис. 14 Код методу перевірки ролі користувача

- **Отримання ідентифікатора користувача** (метод **GetUserId**). Виконується запит до таблиці User, в якому зберігаються дані користувачів, внаслідок якого система передає в запит мобільний номер та отримує від бази даних ID_user, якому він належить, цей ID використовується для виконання іншого функціоналу системи.
- **Відкриття інтерфейсу** (метод **OpenMainForm**) — в залежності від ролі користувача відкривається форма клієнта або працівника, якій передається ідентифікатор (ID_user) та відкрите з'єднання з БД.

2) **Клієнт** (форма **Client_form.cs**). Основна мета — надання клієнтам функціоналу для перегляду, скасування та подачі заявок на ремонтний візит, а також редагування особистих контактних даних і формування звіту.

Основні функції:

- **Отримання особистої інформації** (метод **LoadUserData**). Система виконує запити до таблиць User і Client для завантаження електронної пошти та адреси клієнта за його ідентифікаційним кодом (ID_user). Потім ці дані заповнюються у відповідні текстові поля на сторінці контактної інформації для їх перегляду та редагування.
- **Редагування даних** (**Save_changed_client_data_Click**). Після виклику цієї події клієнт оновлює в контактних даних свій email та адресу на ті, що ввів у відповідні поля, змінюються відповідні атрибути в таблицях User та Client за допомогою SQL-команди **UPDATE**.
- **Додавання заявки на візит** (подія **Add_meeting_Click**), яка, після вибору дати, часу, типу ремонту та опису проблеми виконує:
    - перевірки на дублювання заявки в той самий час, наявність інтервалу не менше 2 годин між ними;

- збереження заявки в таблицю Repair_request із початковим статусом «призначено», за умови що перевірки пройдені успішно.

- **Перегляд заявок** (метод **LoadMeetingsForClient**), що завантажує з БД всі заявки авторизованого клієнта з таблиці Repair_request та показує їх у таблиці dataGridView2.
- **Скасування заявки на візит обслуговування** (подія **cancellation_of_visit_request_Click**). При виникненні цієї події виконується команда **UPDATE** до таблиці Repair_request, яка змінює статус обраної заявки на «скасовано» по її ID.
- **Формування звіту** (**reportViewer1**). Для формування та перегляду звіту по заявкам клієнта за нинішній місяць створено елемент ReportViewer, який містить функціонал для експорту в різні формати.

3) **Працівник служби підтримки** (форма **Worker_form.cs**). Основною метою цього сервісу є надання функціоналу працівнику служби підтримки компаній IT для перегляду візитів та заявок, їх обслуговуванню та зміни статусу, заповненню інформації про виконану роботу, перегляду звітів.

Основні функції:

- **Завантаження журналу заявок** (**метод LoadSchedule**), при виконанні цього методу система завантажує заявки зі статусами «призначено» або «в процесі» та виводить їх у вигляді таблиці для перегляду працівника, додаючи до заявок назву типу проблеми, дані клієнта.
- **Зміна статусу заявки** (**метод UpdateRequestStatus**) — працівник змінює статус заявки на «виконано», «в процесі», «призначено» або «скасовано», залежно від етапу її виконання.

- **Заповнення акта виконаних робіт** (подія **AddCompletedWork_Click**), під час спрацювання якої виїзний працівник обирає заявку з випадаючого списку, описує які роботи по заявці виконав, їх ціну, дату завершення, ID працівника, після чого за допомогою команди **INSERT** дані вносяться в таблицю Completed_work. Аналогічно відбувається подія **AddRepairVisit_Click**, проте там вказується час разом з датою, описується робота виконана лише під час візиту на вказаний час та не записується ціна, адже вона буде в акті виконаних робіт по всім візитам заявки.

На рис. 15 показано блок-схему алгоритму додавання заявки на візит.

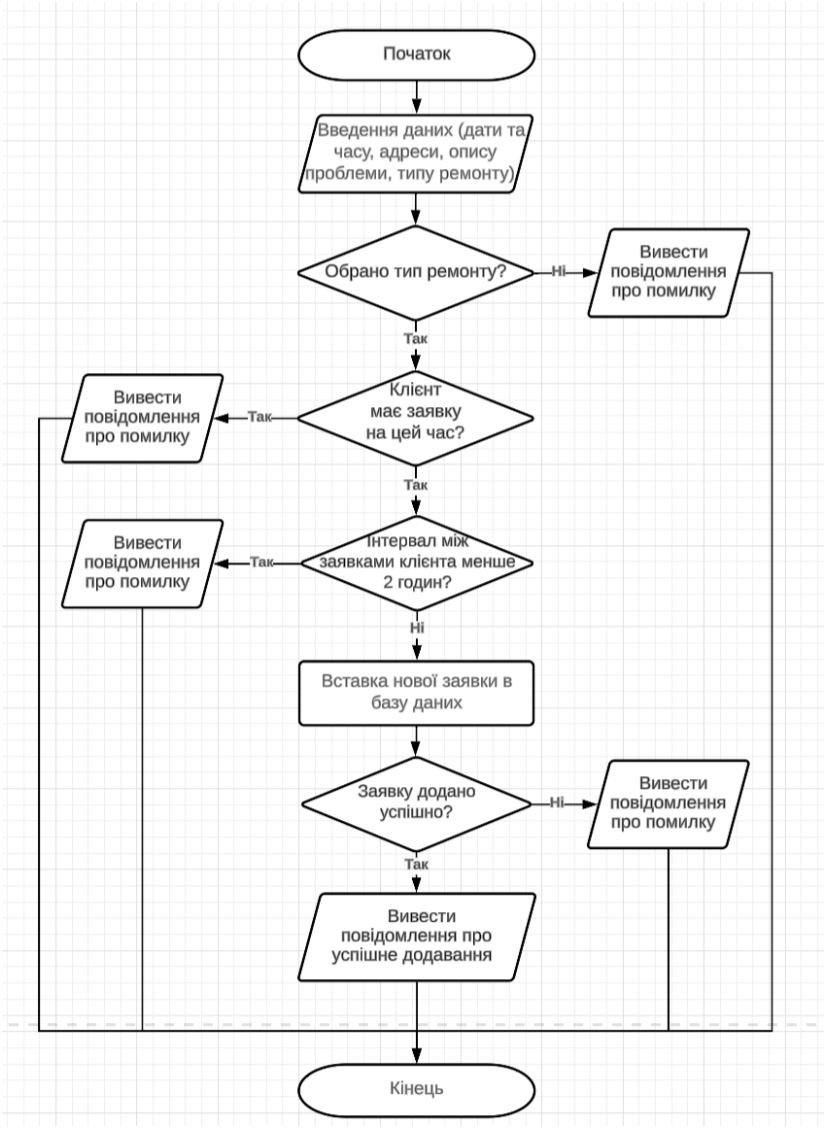


Рис. 15 Блок-схема алгоритму додавання заявки на візит

Код методу додавання заявки на візит знаходиться у додатку Д.

3.5 Проектування і реалізація взаємодії з базою даних

Для взаємодії з базою даних обрано технологію ADO.NET. Вона використовує об’єкти типу SqlConnection, SqlCommand, SqlDataReader для роботи з БД.

Для підключення до бази даних створено файл DBConnection, що містить рядок підключення, що видобувається методом GetConnectionString() для SqlConnection. Потім відкривається з’єднання за допомогою connection.Open().

Клас AuthorizationService містить функціональні методи для SQL-запитів, використовуючи об’єкт SqlCommand. Для запобігання SQL-ін’єкціям використано параметри, наприклад, методом Parameters.AddWithValue() для запиту додається @PhoneNumber.

Для отримання результатів запитів використовується метод ExecuteScalar(), адже він повертає одне значення (наприклад, ідентифікатор користувача).

Потім відкривається відповідна до ролі користувача форма, якій передається активне підключення до бази даних.

Отже, ADO.NET використано для підключення БД, ефективного виконання запитів, роботи з параметрами.

Для створення звітів використано збережені процедури, адже з їх допомогою можна виконувати складні операції відбору та агрегації даних без необхідності їх виконувати у коді програми. Це підвищує продуктивність програми та зменшує навантаження на неї. Також було використано SqlDataAdapter для заповнення DataTable результатами запиту для звіту, адже цей елемент забезпечує ефективну роботу з даними та синхронізацію змін між базою даних і таблицею.

Обрано ReportViewer для виводу даних звіту, сформованого з датасету, який викликає збережену процедуру БД.

3.3.1 Робота з базою даних для клієнта

Підключення до бази даних виконується один раз за допомогою об’єкта SqlConnection. Це підключення використовується у всіх запитах до бази даних.

1) Зчитування даних. Під час завантаження головної форми клієнта виконуються кілька SQL-запитів:

- з таблиці **User** зчитується поле **Phone_number** для поточного користувача за його **ID**;
- з таблиці **Client** отримується поле **Address** за тим же ID користувача;
- з таблиці **Repair_type** витягуються всі типи ремонтів, які заповнюються у відповідний випадаючий список;
- для відображення всіх заявок клієнта виконується запит, що об’єднує дані із таблиць **Repair_request** і **Repair_type** для отримання необхідних даних;
- для можливості скасування заявки відбувається запит для отримання всіх ідентифікаційних номерів призначених заявок цього клієнта та завантаження їх у відповідний випадаючий список.

2) Оновлення даних. При редагуванні клієнтом особистих даних оновлюються поля **Email** у таблиці **User** та **Address** у таблиці **Client** відповідно до змінених значень у полях для вводу. Для цього використовуються два окремі SQL-запити «UPDATE», які передають у параметрах нові значення.

3) Додавання нової заявки. Коли клієнт подає заявку на візит працівника підтримки, програма виконує перевірку, чи немає заявки з тою самою датою та з різницею у часі, меншою за 2 години (щоб уникнути дублювання часу заявок). Якщо перевірка успішна, додається нова заявка в таблицю **Repair_request**. При цьому зберігається дата, адреса, ID користувача, тип ремонту, опис проблеми та встановлюється статус «призначено».

4) Скасування заявки. Якщо клієнт не зможе бути присутнім на візиті, то він його скасовує, вказавши ID заявки, тоді виконується запит **UPDATE**, що оновлює статус на «скасовано».

3.3.2 Робота з базою даних для працівника підтримки

У Worker_form використовується зв’язок з базою даних SQL Server для отримання, оновлення та обробки ремонтних заявок. Об’єкт SqlConnection передається у конструктор форми для роботи з БД.

1) Завантаження персональної інформації

При завантаженні форми виконується SQL-запит, який використовує **INNER JOIN** між таблицями [User], Support_worker, щоб за ID користувача отримати всю необхідну інформацію. Дані відображаються у відповідних полях форми.

2) Вивід поточних заявок

Форма автоматично завантажує список заявок (що мають статус «призначено» або «в процесі») за допомогою об’єднання Repair_request і Repair_type. Для кожної заявки відображається:

- ID заявки,
- дата й час ремонту,
- адреса,
- тип ремонту,
- опис проблеми.

Цей список призначений для того, щоб працівник міг обрати заявку для виконання.

3) Зміна статусу заявки

Коли працівник обирає заявку з випадаючого списку та змінює її статус, то виконується SQL-запит UPDATE.

4) Завершення виконання

Після завершення ремонту працівник за допомогою INSERT вносить дані щодо візиту та виконаних робіт, а потім обирає заявку і змінює статус на «проведено», використавши у запиті оператор UPDATE.

На рис. 16 показано блок-схему алгоритму відбору даних для формування звіту.

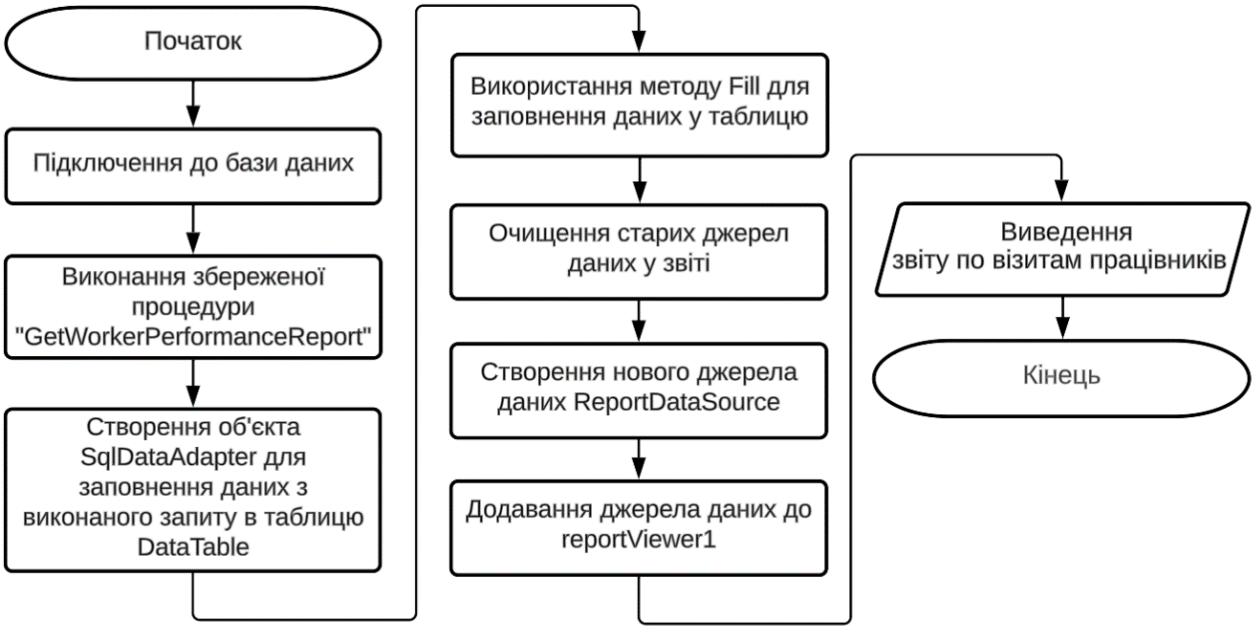


Рис. 16 Блок-схема алгоритму відбору даних для формування звіту

На рис. 17 зображено код методу формування звіту про виконану роботу працівників.

```
private void Form1_Load(object sender, EventArgs e)
        {
            SqlCommand cmd = new SqlCommand("GetWorkerRepairReport", connection);
            cmd.CommandType = CommandType.StoredProcedure;
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            reportViewer1.LocalReport.DataSources.Clear();
            // Динамічний шлях до файлу звіту
            string reportPath = Path.Combine(Application.StartupPath, "Reports", "Report1.rdlc");
            reportViewer1.LocalReport.ReportPath = reportPath;
            ReportDataSource rds = new ReportDataSource("DataSet_1", dt);
            reportViewer1.LocalReport.DataSources.Add(rds);
            reportViewer1.RefreshReport();
        }
```

Рис. 17 Код методу формування звіту про виконану роботу працівників

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Діаграма розгортання

На рис. 18 зображено діаграму розгортання інформаційної системи, яка показує взаємодію між клієнтським застосунком та хмарним сервером бази даних. Користувач, працюючи в програмі «ІС обліку обслуговування клієнтів компаній ІТ», яка встановлена у нього на комп’ютері, зв’язується з хмарним SQL-сервером (Google Cloud SQL) через мережевий протокол TCP/IP для обміну даними. Сервер обробляє запити до бази даних, забезпечує централізоване зберігання та надання дистанційного доступу до інформації.

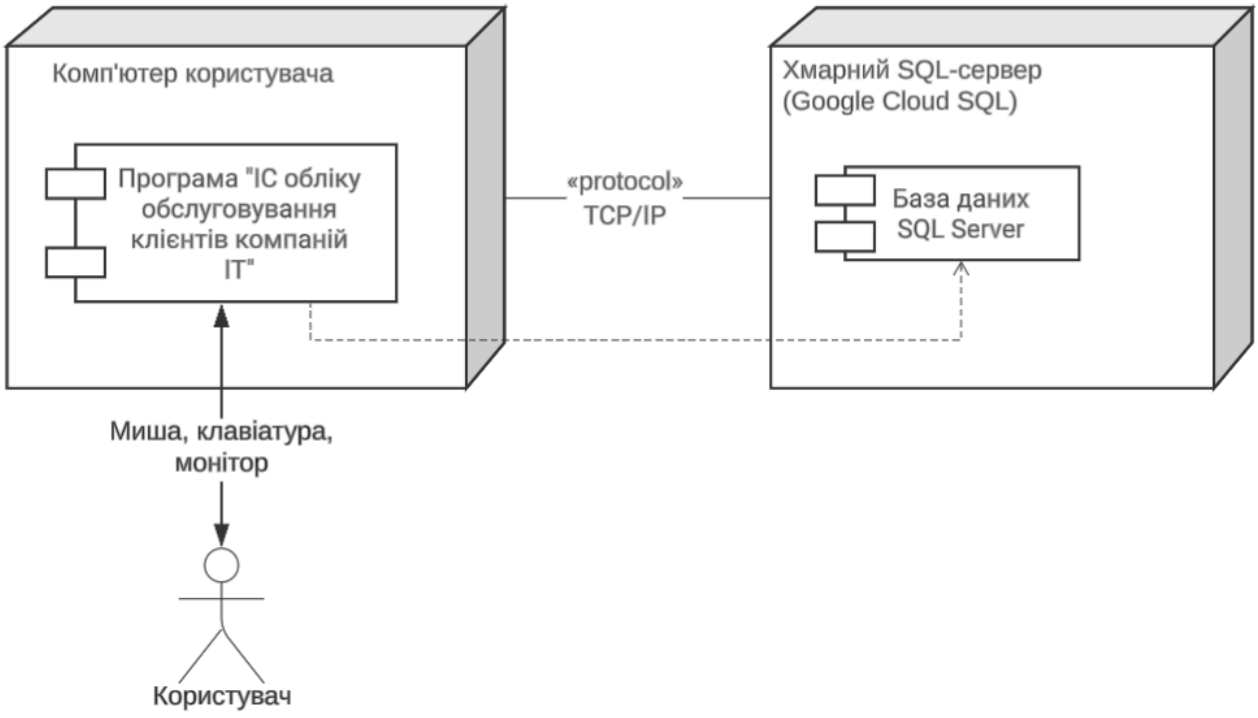


Рис. 18 Діаграма розгортання

4.2 Вимоги до апаратного та програмного забезпечення

4.2.1 Апаратні вимоги. Клієнтська частина вимагає таке обладнання.

- Процесор: Intel Core i3 або вище.
- Оперативна пам’ять: мінімум 4 ГБ.
- Диск: не менше 500 МБ вільного простору.
- Роздільна здатність екрану: мінімум 1366x768.
- Мережеве підключення: стабільне інтернет-з’єднання для зв’язку з сервером.

Серверна частина (SQL Server) потребує наступне обладнання.

- Процесор: Intel Core i5 або потужніший.
- Оперативна пам’ять: від 8 ГБ.
- Диск: SSD, мінімум 10 ГБ вільного місця.
- Операційна система: Windows Server 2022.
- Мережева карта: Gigabit Ethernet або краще.

4.2.2 Програмні вимоги. Вимоги до клієнтської частини такі.

- Операційна система: Windows 10 або 11.
- .NET Framework: мінімальна версія 4.6.
- Report Viewer: Microsoft Report Viewer 2015 або новіший (встановлюється всередині інсталяційного пакету з іншими бібліотеками).
- Драйвери: SQL Server Native Client (для доступу до БД).

Вимоги до серверної частини такі.

- SQL Server 2022 (Express / Standard / Enterprise).
- Відкритий доступ по TCP/IP.
- Firewall: відкритий порт 1433 для SQL Server.
- Конфігурація бази даних: створена база даних RepairServiceDB, всі таблиці, збережені процедури, тригери та користувачі налаштовані відповідно до логіки роботи програми.

4.3 Склад інсталяційного пакету

На рис. 19 показано інсталяційний пакет, створений за допомогою Microsoft Visual Studio Installer Projects. При запуску інсталятора буде розпаковано проект, його файли та бібліотеки, створено ярлик запуску програми на робочому столі.

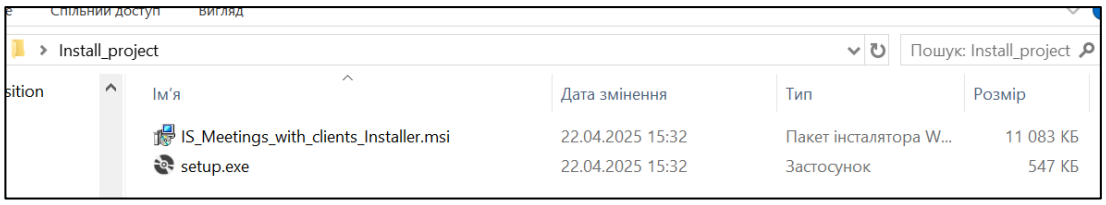


Рис. 19 Інсталяційний пакет системи

На рис. 20 показано вміст папки, всередину якої було інстальовано програму.

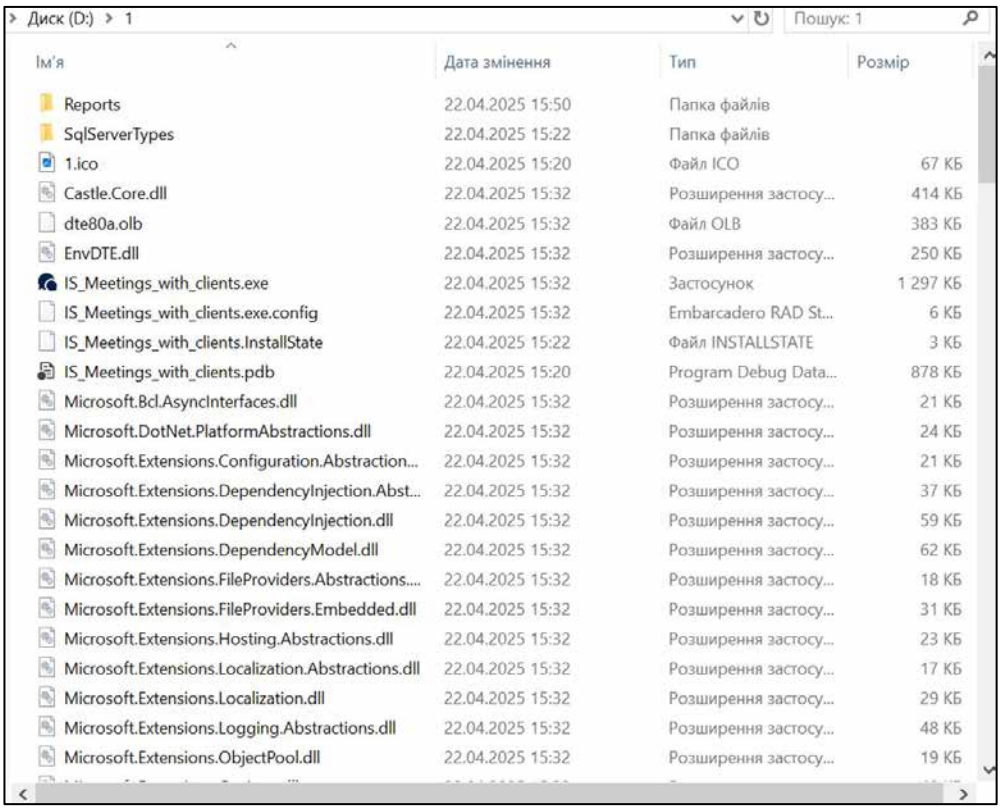


Рис. 20 Вигляд папки проекту після інсталяції

Після розпакування проекту з файлу «setup.exe» або «IS_meetings_with_clients_Installer.msi» в обрану користувачем папку інсталятор видобуває файли проєкту, такі як:

- звіти — файли формату .rdlc, що знаходяться в папці Reports;

- бібліотеки формату .dll, що використовувалися при створенні проекту та тестуванні його модулей: Microsoft.DotNet.*, Microsoft.Extensions.*, та інші;
- Microsoft.Reporting.* — бібліотеки для створення звітів, що мають тип файлу .dll;
- SqlServerTypes (папка з бібліотеками .dll для зв’язку з SQL Server);
- IS_Meetings_with_clients.exe — основний виконуючий файл програми, що запускає її на виконання.
- 1.ico — іконка вигляду програми.

4.4 Тестування системи

Тестування програмного забезпечення — це процес перевірки відповідності системи вимогам та коректності її роботи в різних умовах.

Основними її типами є:

- модульне тестування — тестування окремих частин системи;
- інтеграційне тестування — перевірка взаємодії між модулями;
- системне тестування — перевірка роботи системи загалом, як єдиного механізму;
- приймальне тестування (User Acceptance Testing) — перевірка відповідності системи приймальним критеріям замовника [22];
- мануальне тестування — ручна перевірка функціоналу без використання спеціальних інструментів, які автоматизують цей процес;
- автоматизоване тестування — застосування спеціального ПЗ для автоматичного виконання тестів;
- функціональне тестування — перевірка роботи усіх функцій відповідно до специфікації;

- нефункціональне тестування — оцінка продуктивності, зручності використання, сумісності та інших показників [23].

Було використано мануальне тестування, тобто вручну перевірено правильність роботи функцій програми. На рис. 21 користувач вводить свій логін та пароль, при успішній авторизації відкривається форма клієнта або працівника (відповідно до його ролі).

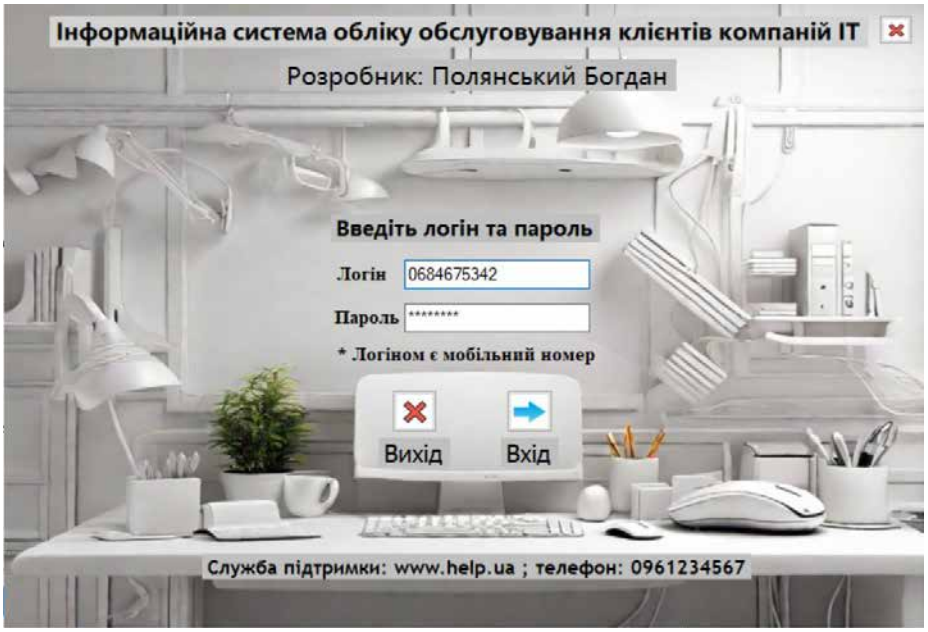


Рис. 21 Авторизація користувача

На рис. 22 показано перегляд та редагування контактних даних клієнта. При натисненні кнопки «Зберегти» зміни вносяться до БД.

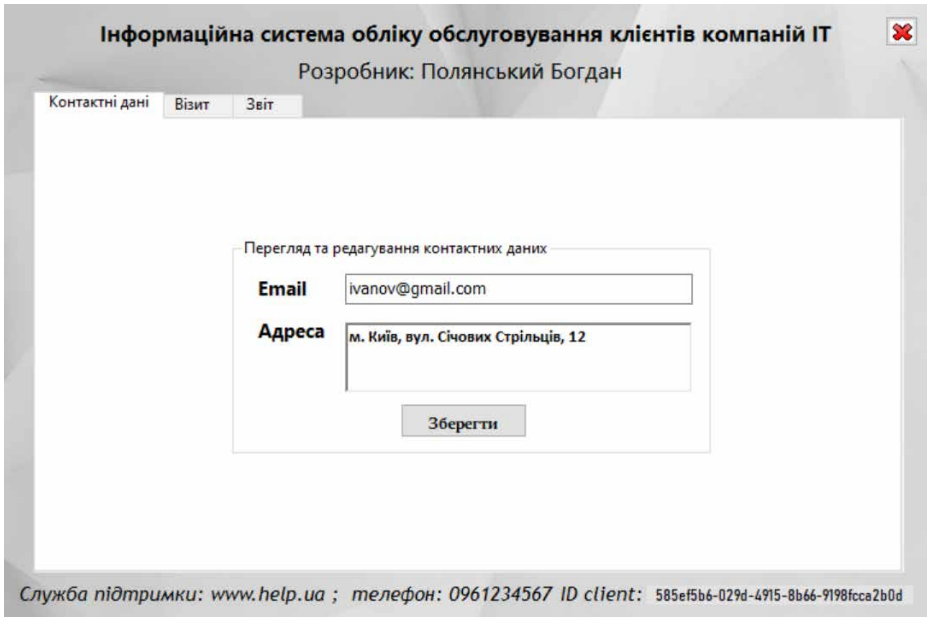


Рис. 22 Перегляд та редагування контактних даних клієнта

На рис. 23 зображено запис на візит, під час якого клієнт подає заявку на обслуговування працівнику служби підтримки, обравши дату та час, адресу, тип ремонту і описавши проблему.

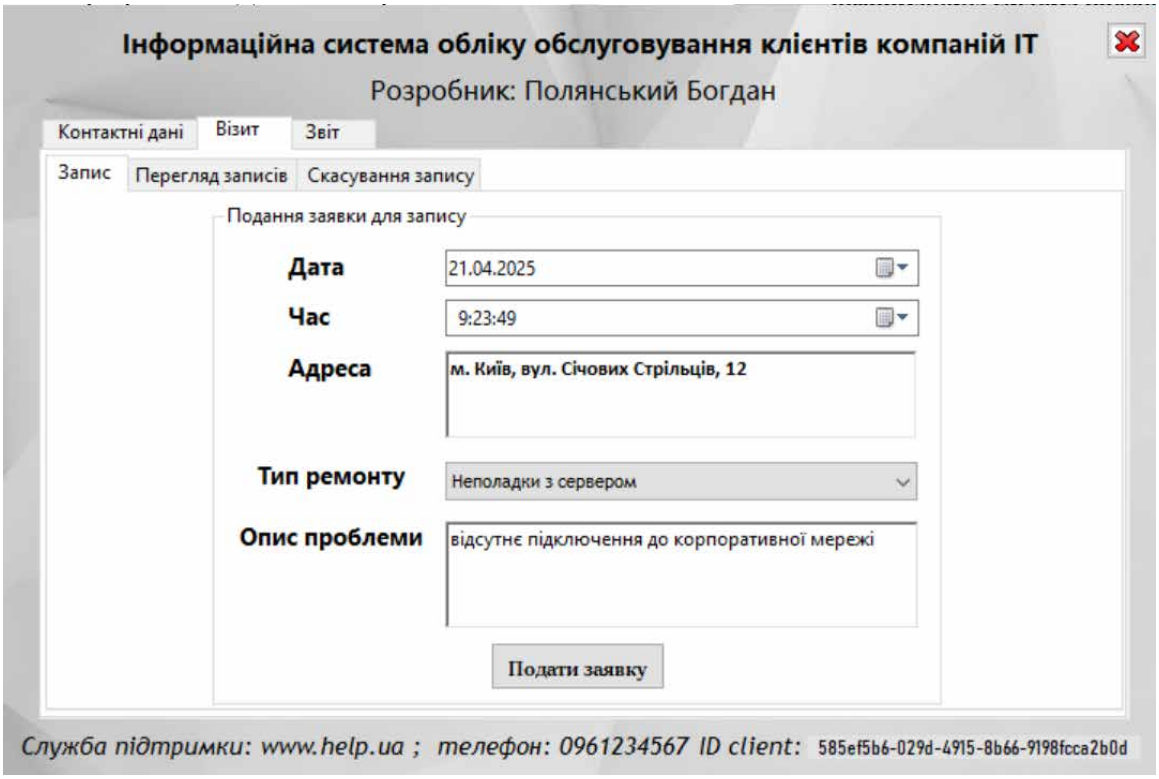


Рис. 23 Запис на візит

На рис. 24 показано перегляд власних записів на візит клієнта.

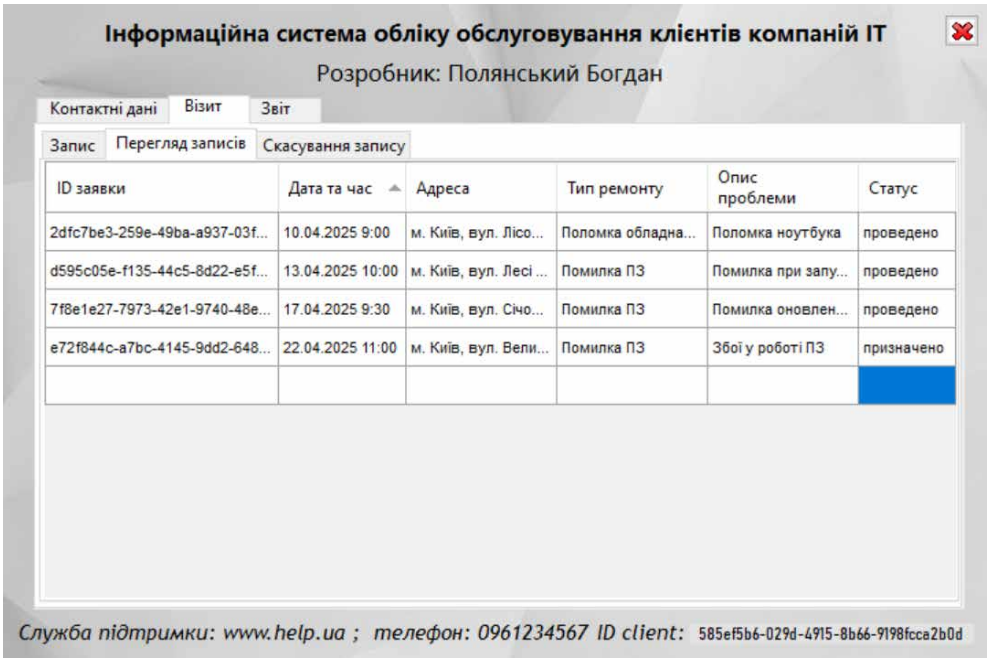


Рис. 24 Перегляд записів на візит авторизованого клієнта

На рис. 25 показано скасування клієнтом запису на візит, для виконання якого він обирає з випадаючого списку код візиту (який там буде, в більшості випадків, один, адже дві заявки підряд зазвичай не назначають), якщо їх там декілька, спочатку виконується перегляд записів на візит, щоб дізнатися код потрібної заявки, а потім вибір її з списку та скасування при натисненні відповідної кнопки.

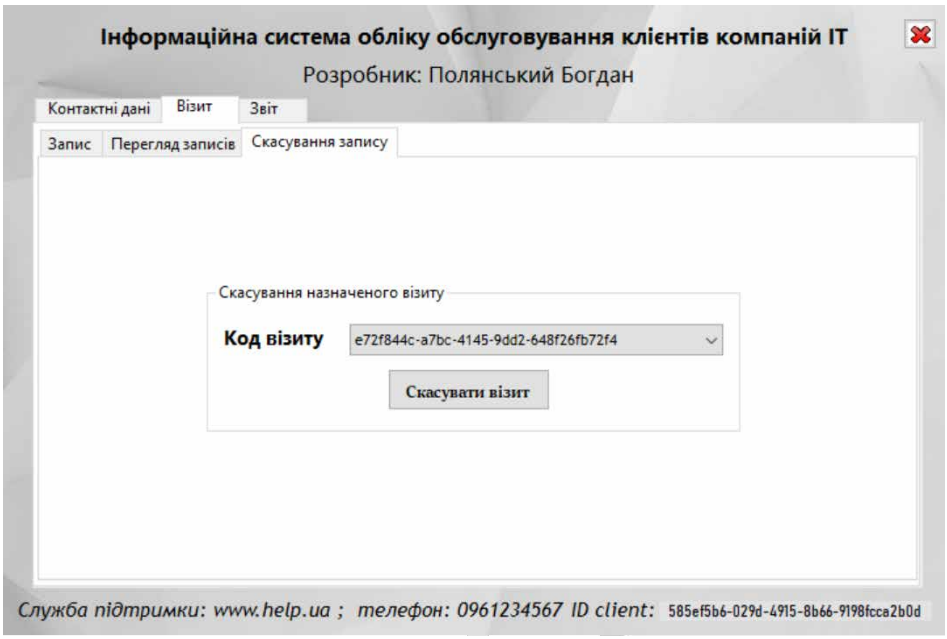


Рис. 25 Скасування запису на візит

На рис. 26 зображено звіт по візитам клієнта за останній місяць.

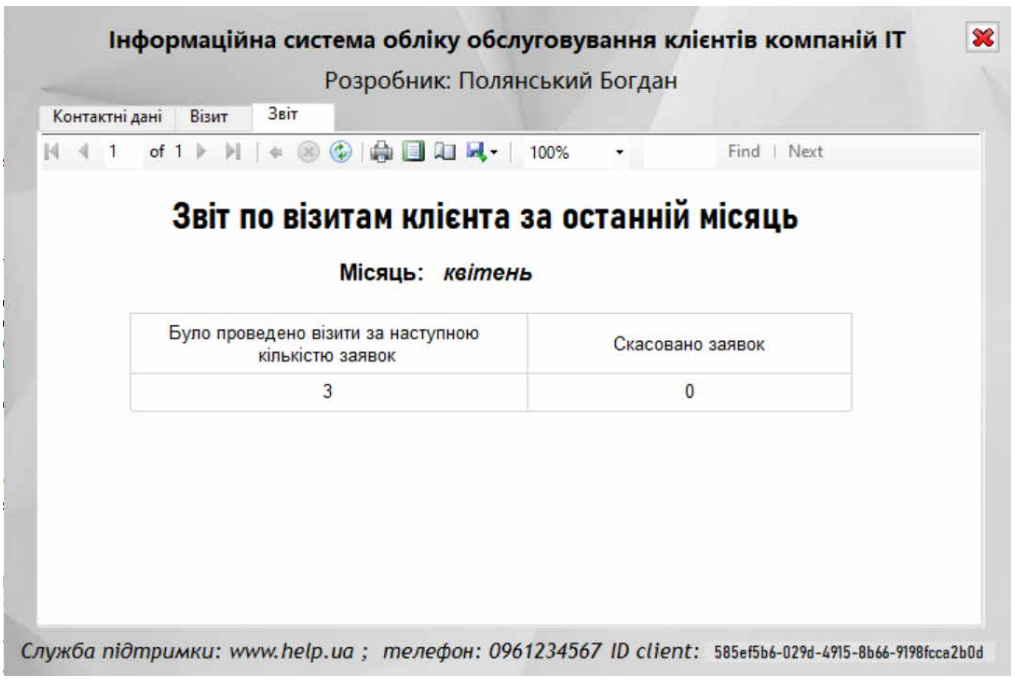


Рис. 26 Звіт по візитам клієнта за останній місяць

На рис. 27 зображено перегляд активних заявок на візит працівником, який відбувається для майбутнього проведення обслуговування, внесення даних по візитам та актам виконаних робіт, встановленя статусу заявки у відповідній вкладці.

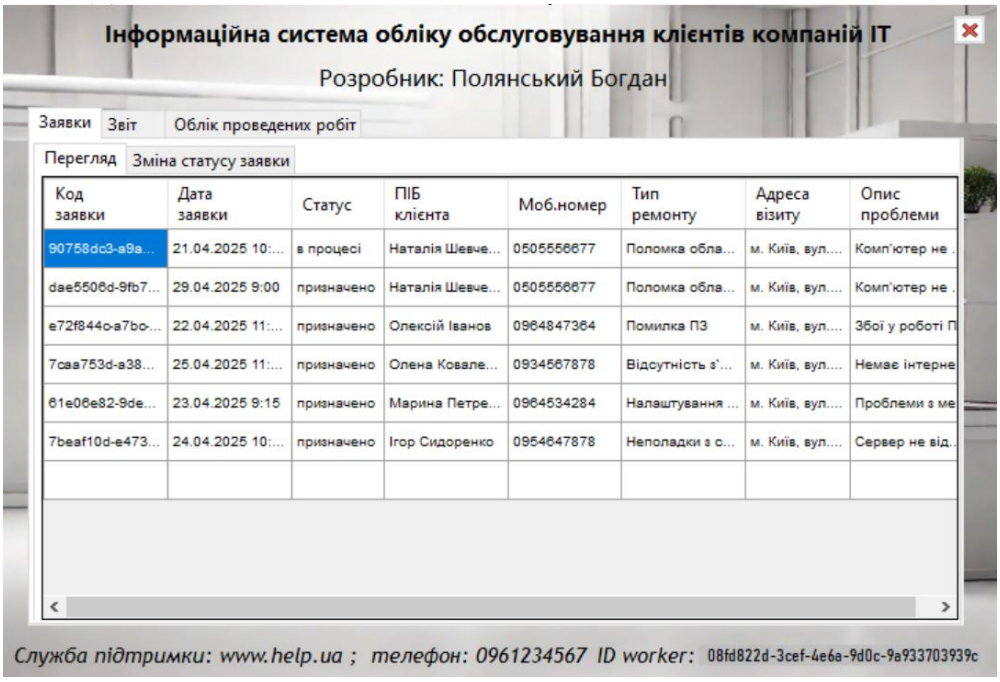


Рис. 27 Перегляд заявок на обслуговування працівником

На рис. 28 зображено зміну статусу заявки працівником під час її опрацювання.

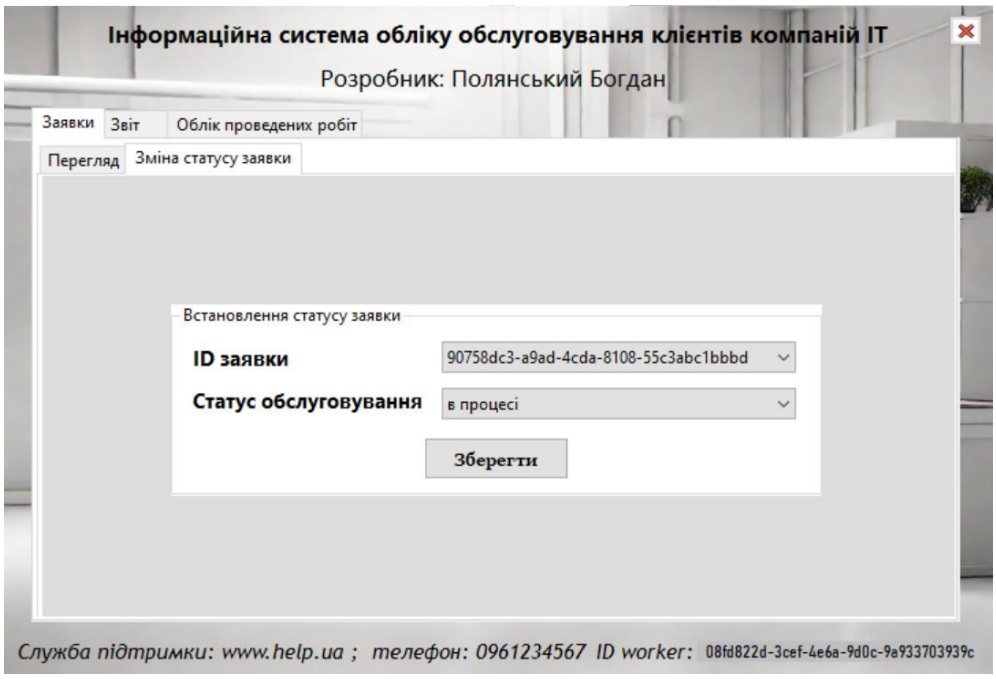


Рис. 28 Зміна статусу заявки працівником

На рис. 29 зображено звіт по роботі працівників. При натисканні на кнопку з іконкою збереження документу відбувається експорт звіту у формат Exel, PDF, Word.

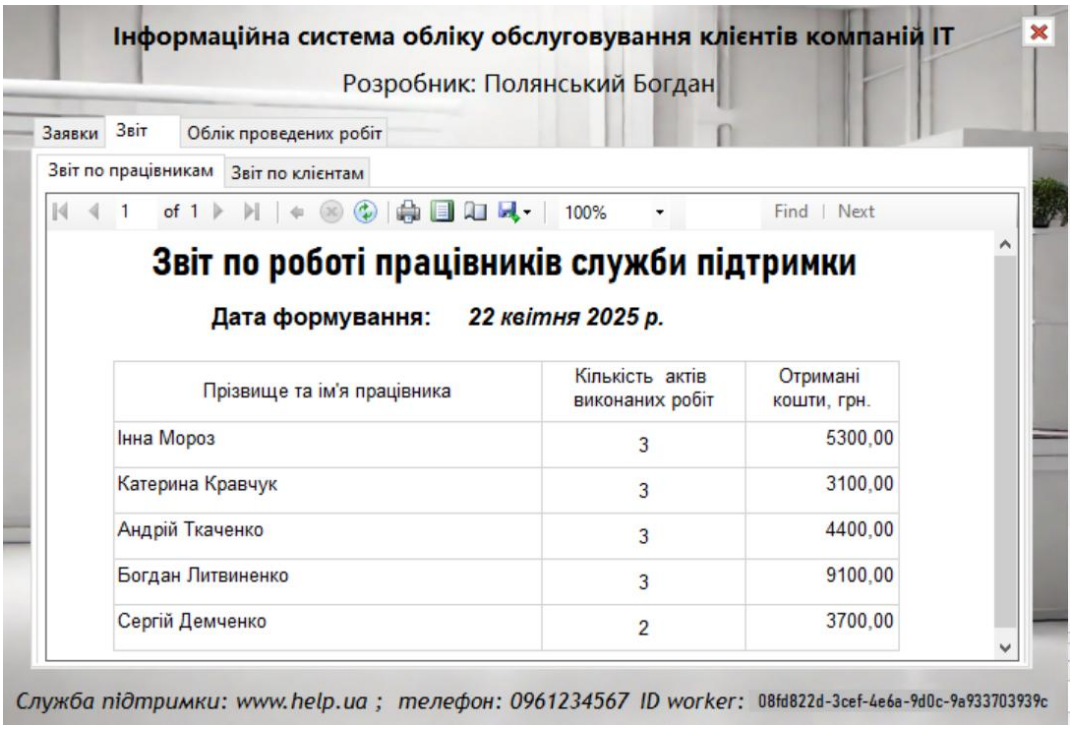


Рис. 29 Звіт по роботі працівників служби підтримки

На рис. 30 працівник обирає місце збереження звіту та може змінити назву файлу.

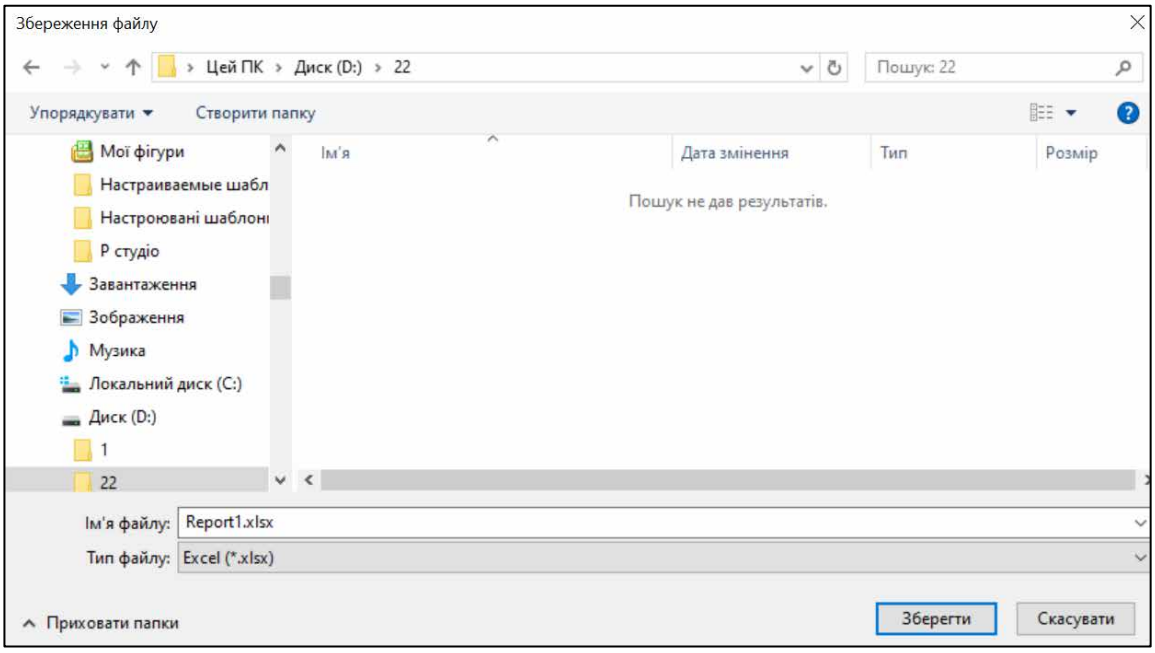
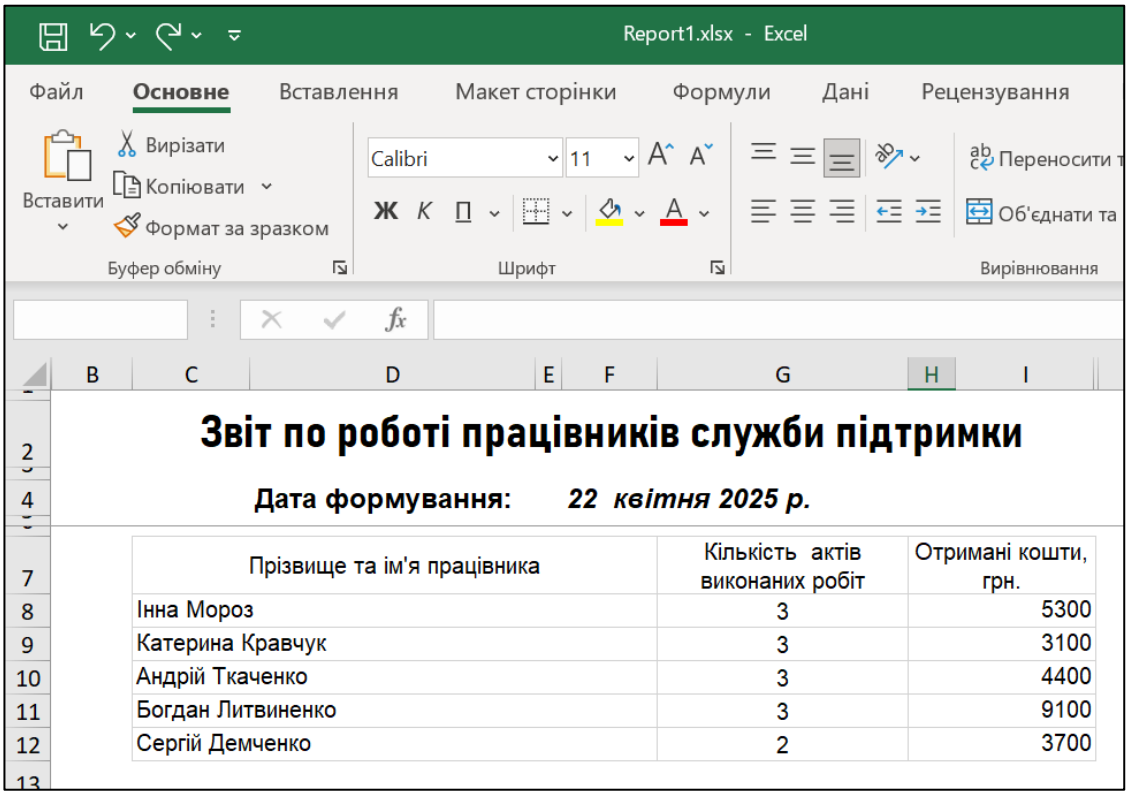


Рис. 30 Вікно для вибору місця збереження звіту

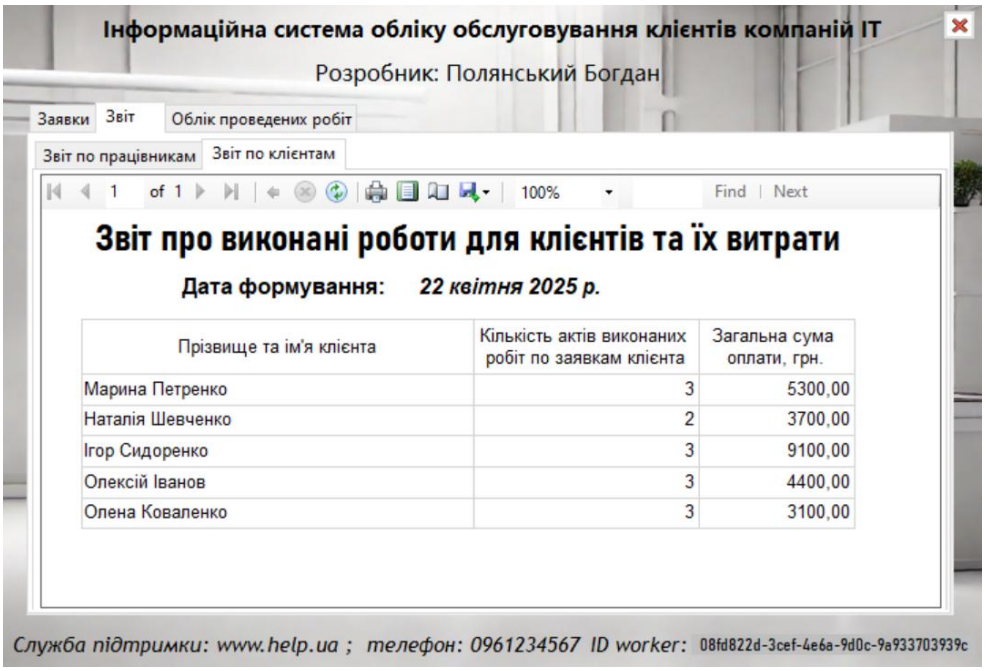
Працівник, перейшовши до місця зберігання файлу та відкривши його, переглядає експортований звіт у форматі .xlsx, що зображений на рис. 31. Аналогічно відбувається експорт інших звітів.



Прізвище та ім'я працівника	Кількість актів виконаних робіт	Отримані кошти, грн.
Інна Мороз	3	5300
Катерина Кравчук	3	3100
Андрій Ткаченко	3	4400
Богдан Литвиненко	3	9100
Сергій Демченко	2	3700

Рис. 31 Експортований звіт у Exel

На рис. 32 зображено звіт про виконані роботи для клієнтів та їх витрати.



Прізвище та ім'я клієнта	Кількість актів виконаних робіт по заявкам клієнта	Загальна сума оплати, грн.
Марина Петренко	3	5300,00
Наталія Шевченко	2	3700,00
Ігор Сидоренко	3	9100,00
Олексій Іванов	3	4400,00
Олена Коваленко	3	3100,00

Рис. 32 Звіт про виконані роботи для клієнтів і їх витрати

На рис. 33 зображено додавання даних проведеного візиту працівником, в якому він обирає ID заявки, система автоматично заповнює його ID працівника, а він вносить дані дати та часу, описує виконану роботу і нажимає на відповідну кнопку збереження. При успішному виконанні цієї дії показується вспливаюче вікно про додавання візиту, інакше — помилка.

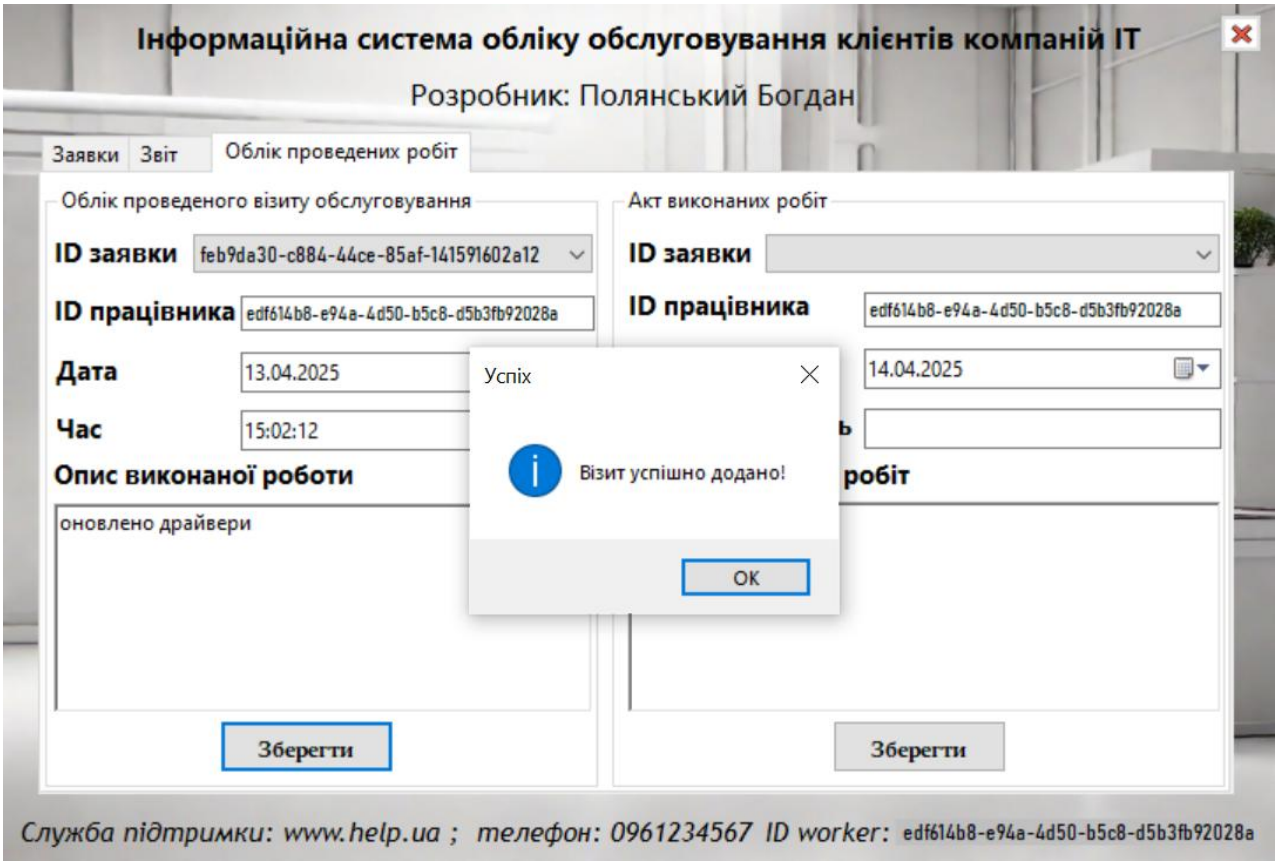


Рис. 33 Облік проведеного візиту обслуговування працівником

На рис. 34 показано додавання акта виконаних робіт по заявці клієнта, коли було вирішено його проблему під час візитів. Працівник виконує аналогічні дії до обліку проведеного візиту: обирає код заявки та вказує дату закінчення виконання робіт, описує виконані роботи по візитах цієї заявки, а програма автоматично вказує його код працівника. При успішному внесенню даних в БД виводиться повідомлення про виконання операції.

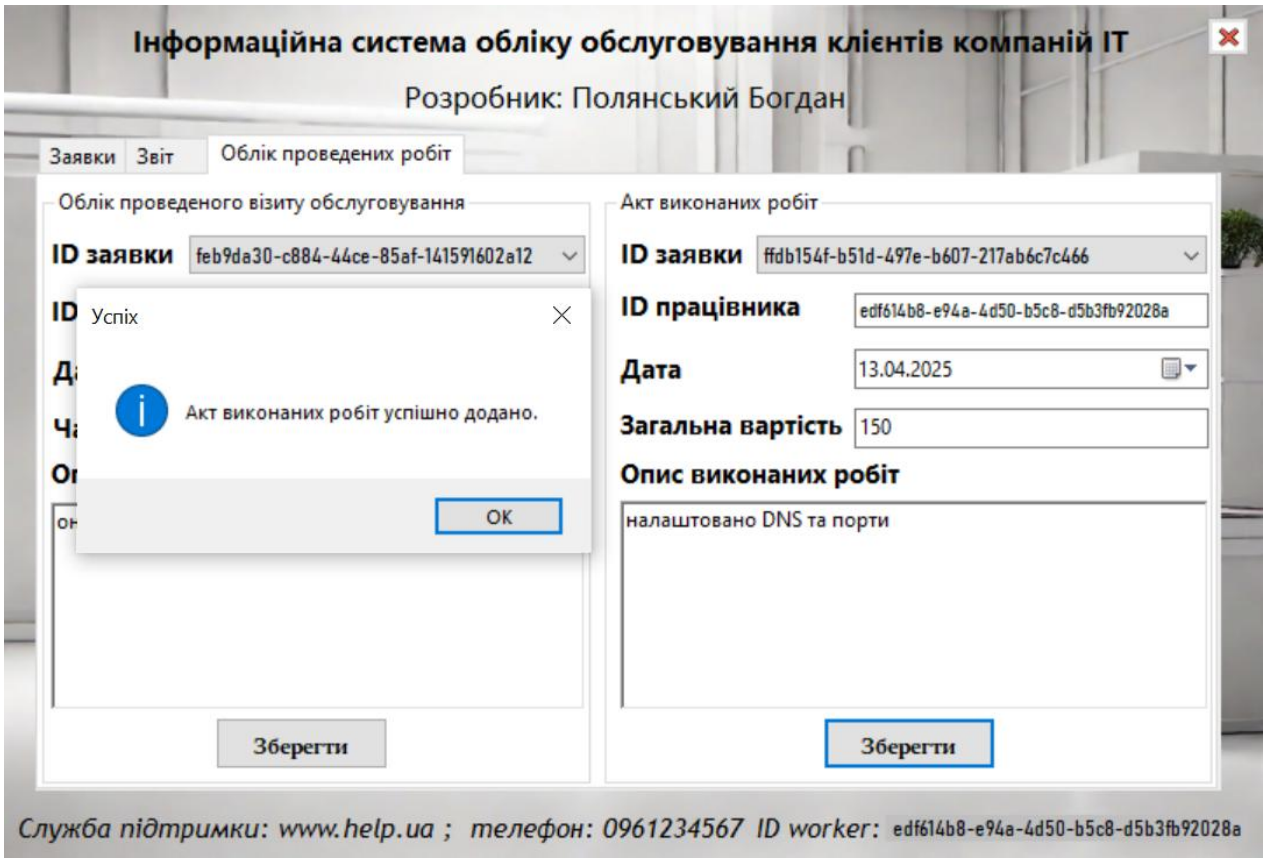


Рис. 34 Додавання акта виконаних робіт

ВИСНОВКИ

У межах дипломної роботи було описано дизайн та архітектуру застосунку, розроблено інформаційну систему обліку обслуговування клієнтів компаній IT, з метою автоматизації основних процесів, пов’язаних із плануванням та обліком візитів технічних працівників. Розроблена система суттєво спрощує керування заявками, ведення обліку візитів та виконаної роботи, створення звітів, а також оптимізує роботу працівників служби підтримки.

Результати виконання проєкту свідчать про досягнення мети та успішне виконання поставлених технічних завдань. Увесь розроблений функціонал системи відповідає сучасним вимогам до подібного програмного забезпечення, а також має суттєві переваги в порівнянні з існуючими системами, такі як: зручний користувацький інтерфейс, швидкий доступ до даних, збереження історії обслуговування та інтеграцію з хмарним сховищем.

Дана система враховує особливості та специфіку технічного обслуговування в IT-сфері та дозволяє ефективно керувати візитами служби технічної підтримки, а також зберігати акти виконаних робіт та інформацію про проведені візити.

Технічними нововведеннями є інтеграція хмарного сервісу Google Cloud SQL для надання віддаленого доступу до даних, що забезпечує їх надійне збереження та доступність, а також використання механізму для зручної генерації звітів за даними візитів та заяв на обслуговування у реальному часі. Це дозволяє істотно підвищити техніко-економічну ефективність, зменшити необхідну кількість часу на адміністративну роботу та поліпшити якість обслуговування.

Розроблена система може бути впроваджена в невеликі та середні бізнеси, які надають технічну підтримку, а також для інших компаній, для яких важливим є ведення обліку виконаних робіт та виїзних візитів. На основі отриманого

досвіду рекомендовано продовжити розвиток системи: створити мобільний застосунок, додати модуль аналітики та GPS-трекер, розширити функціональні можливості для зв’язку з клієнтами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IT-підтримка для бізнесу: що це, види та переваги - Netwave. *Netwave*. URL: https://netwave.ua/blog/it-pidtrimka-shho-cze-etapi-ta-perevagi-dlya-biznesu-copy/ (дата звернення: 25.05.2025).
2. Maxym Z. Діаграма послідовності (Sequence Diagrams). *Maxym Zosym*. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 25.05.2025).
3. Елементи UML. *KDE Documentation*. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html (дата звернення: 25.05.2025).
4. SimplyBook.me - Безкоштовна Система Попереднього Запису. *SimplyBook.me - Free Appointment Booking System*. URL: https://simplybook.me/uk/ (дата звернення: 25.05.2025).
5. Програма онлайн-запису клієнтів та управління бізнесом. *Програма онлайн-запису та CRM управління бізнесом - EasyWeek*. URL: https://easyweek.com.ua/ (дата звернення: 25.05.2025).
6. Maxym Z. Моделювання даних (Data Modelling). *Maxym Zosym*. URL: https://www.maxzosim.com/data-modelling/ (дата звернення: 25.05.2025).
7. Схеми бази даних: компоненти, типи та проєктування. *FoxmindEd*. URL: https://foxminded.ua/skhemy-bazy-danyh/ (дата звернення: 25.05.2025).
8. Логічна модель даних - data-life-ua. *data-life-ua*. URL: https://data-life-ua.com/db/lohichna-model-danykh/ (дата звернення: 25.05.2025).
9. Реляційні бази даних. *Relational databases*. URL: https://rdb.dp.ua/uk/chapter_03 (дата звернення: 25.05.2025).

10. Система управління базами даних: сучасні тенденції у розробці. *FoxmindEd*. URL: https://foxminded.ua/systema-upravlinnia-bazamy-danykh/ (дата звернення: 25.05.2025).
11. Реляційні бази даних: Що це таке і як вони працюють. *ALexHost SRL*. URL: https://alexhost.com/uk/faq/relyaczijni-bazy-danyh-shho-cze-take-i-yak-vony-praczyuyut/ (дата звернення: 25.05.2025).
12. Програмний продукт Microsoft Access LTSC 2024 (DG7GMGF0PN5J-0002). *ERC Ukraine*. URL: https://erc.ua/ware/381278-microsoft-access-ltsc-2024/ (дата звернення: 25.05.2025).
13. *Львівське ВПУ ІКТ – Освітній заклад із сімдесятирічною історією*. URL: https://lvpuikt.lviv.ua/wp-content/uploads/2020/04/Робота-в-Microsoft-Access-2010.pdf (дата звернення: 25.05.2025).
14. Системи управління базами даних СУБД для малого та середнього бізнесу. *Kyivstar Business Hub – корпоративний блог для бізнесу*. URL: https://hub.kyivstar.ua/articles/sistemi-upravlinnya-bazami-danih-dlya-malogo-ta-serednogo-biznesu-yak-vibrati (дата звернення: 25.05.2025).
15. Explore Top Oracle Database Features. *Oracle*. URL: https://www.oracle.com/ua/database/features/ (дата звернення: 25.05.2025).
16. Порівняння СУБД My SQL, PostgreSQL та MS SQL Server. *Курси бізнес аналітики | Україна | DataBI*. URL: https://data-b-i.com/uk/article/porivnyannya-subd-mysql-postgresql-mssqlserver.html (дата звернення: 25.05.2025).
17. Клієнт-серверна архітектура. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 25.05.2025).
18. Що таке хмарні технології? Переваги та недоліки хмарних сервісів | EDIN. *EDIN*. URL: https://edin.ua/shho-take-xmarni-texnologiï-i-navishho-voni-potribni/ (дата звернення: 25.05.2025).

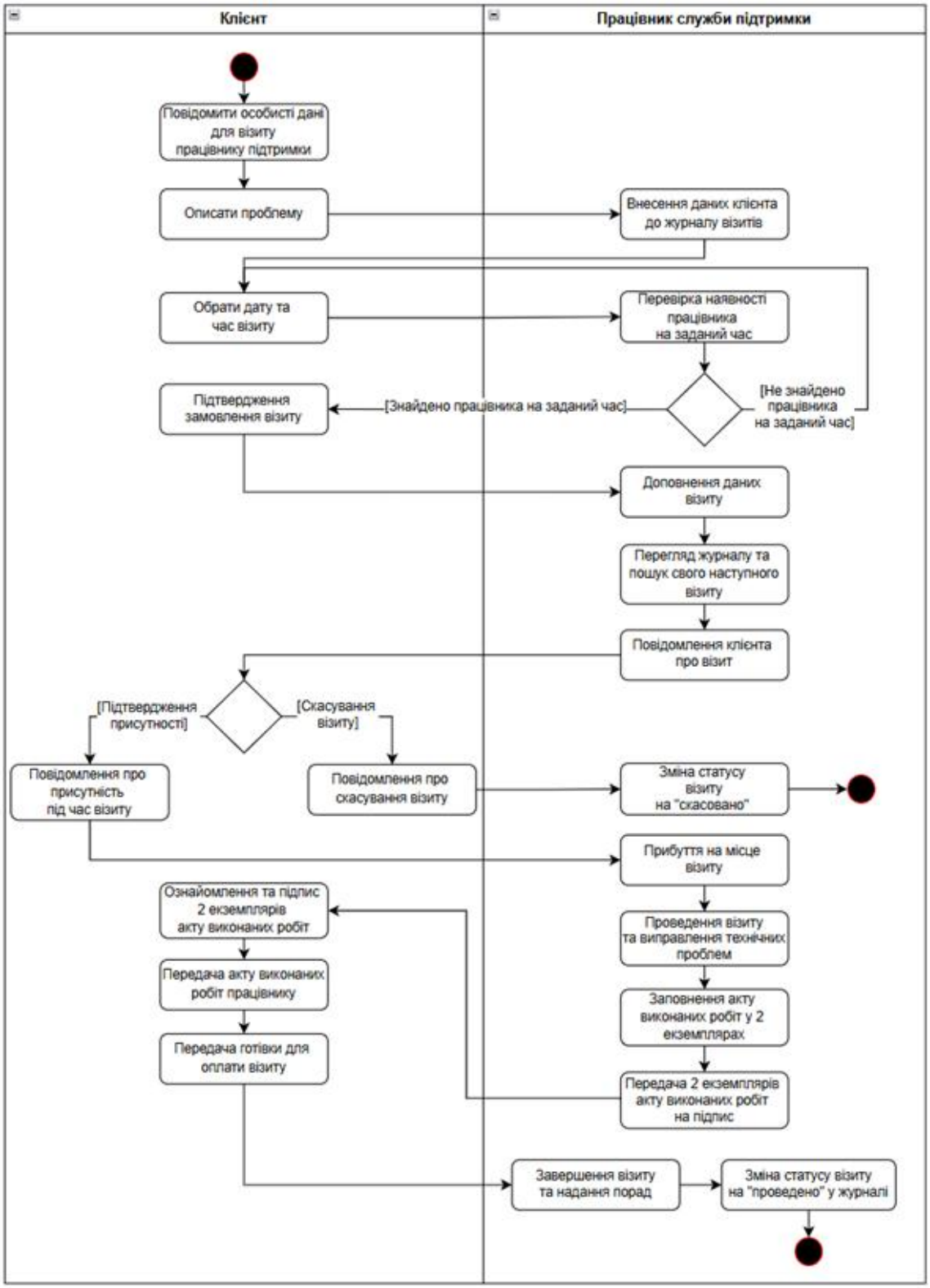
19. Збережені процедури | e-helper.com.ua. *e-helper.com.ua*. URL: https://e-helper.com.ua/node/40 (дата звернення: 25.05.2025).
20. Hushchyn D. Тригери для моніторингу бази даних. *Kovelpost*. URL: https://kovelpost.com/blogs/11698 (дата звернення: 25.05.2025).
21. Lindemulder G., Kosinski M. What Is Role-Based Access Control (RBAC)? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/rbac (дата звернення: 25.05.2025).
22. User Acceptance Testing (UAT) - приймальне тестування та його цілі. *Highload.tech - медіа для розробників*. URL: https://highload.tech/uk/user-acceptance-testing-uat-prijmalne-testuvannya-ta-jogo-tsili/#i (дата звернення: 25.05.2025).
23. Popeliushko N. Всі відомі типи тестування (100+ штук). Nataliia Popeliushko. *Dou*. URL: https://dou.ua/forums/topic/40666/ (дата звернення: 25.05.2025).

ДОДАТОК А

Діаграма прецедентів предметної області

ДОДАТОК Б

Діаграма діяльності предметної області



ДОДАТОК В

Коди створення об’єктів бази даних

Сторінок 7

Київ-2025

Код створення БД і таблиць

```
CREATE DATABASE RepairServiceDB;
GO
USE RepairServiceDB;
GO
-- Видалення таблиць у правильному порядку
IF OBJECT_ID('Completed_work', 'U') IS NOT NULL DROP TABLE Completed_work;
IF OBJECT_ID('Repair_visit', 'U') IS NOT NULL DROP TABLE Repair_visit;
IF OBJECT_ID('Repair_request', 'U') IS NOT NULL DROP TABLE Repair_request;
IF OBJECT_ID('Repair_type', 'U') IS NOT NULL DROP TABLE Repair_type;
IF OBJECT_ID('Support_worker', 'U') IS NOT NULL DROP TABLE Support_worker;
IF OBJECT_ID('Client', 'U') IS NOT NULL DROP TABLE Client;
IF OBJECT_ID('Office', 'U') IS NOT NULL DROP TABLE Office;
IF OBJECT_ID('[User]', 'U') IS NOT NULL DROP TABLE [User];
GO
-- Таблиця користувачів
CREATE TABLE [User] (
    ID_user UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
    Full_name VARCHAR(100) NOT NULL,
    Phone_number VARCHAR(20),
    Email VARCHAR(MAX)
);
-- Таблиця офісів
CREATE TABLE Office (
    ID_office UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
    Office_address VARCHAR(MAX) NOT NULL
);
-- Таблиця клієнтів
CREATE TABLE Client (
    ID_user UNIQUEIDENTIFIER PRIMARY KEY,
    Address VARCHAR(MAX) NOT NULL,
    FOREIGN KEY (ID_user) REFERENCES [User](ID_user) );
```

```
-- Таблиця працівників служби підтримки
CREATE TABLE Support_worker (
    ID_user UNIQUEIDENTIFIER PRIMARY KEY,
    Position NVARCHAR(255) NOT NULL,
    ID_office UNIQUEIDENTIFIER NOT NULL,
    FOREIGN KEY (ID_user) REFERENCES [User](ID_user),
    FOREIGN KEY (ID_office) REFERENCES Office(ID_office)
);
-- Таблиця типів ремонту
CREATE TABLE Repair_type (
    ID_repair_type UNIQUEIDENTIFIER PRIMARY KEY,
    Repair_name VARCHAR(50) NOT NULL,
    [Description_of_types] TEXT
);
-- Таблиця заявок на ремонт
CREATE TABLE Repair_request (
    ID_request UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
    Request_date DATETIME NOT NULL,
    Visit_address VARCHAR(MAX) NOT NULL,
    ID_user UNIQUEIDENTIFIER NOT NULL,
    ID_repair_type UNIQUEIDENTIFIER NOT NULL,
    Problem_description TEXT NOT NULL,
    [Status] VARCHAR(15) NOT NULL,
    FOREIGN KEY (ID_user) REFERENCES Client(ID_user),
    FOREIGN KEY (ID_repair_type) REFERENCES Repair_type(ID_repair_type)
);
-- Таблиця візитів на ремонт
CREATE TABLE Repair_visit (
    ID_visit UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
    ID_user UNIQUEIDENTIFIER NOT NULL,
    Visit_date DATETIME NOT NULL,
    Work_description Text,
    ID_request UNIQUEIDENTIFIER NOT NULL,
```

```
    FOREIGN KEY (ID_user) REFERENCES Support_worker(ID_user),
    FOREIGN KEY (ID_request) REFERENCES Repair_request(ID_request)
);
-- Таблиця завершених робіт
CREATE TABLE Completed_work (
    ID_request UNIQUEIDENTIFIER PRIMARY KEY,
    Completion_date DATE NOT NULL,
    Work_description TEXT NOT NULL,
    Total_cost DECIMAL(10,2) NOT NULL,
    ID_user UNIQUEIDENTIFIER NOT NULL,
    FOREIGN KEY (ID_request) REFERENCES Repair_request(ID_request),
    FOREIGN KEY (ID_user) REFERENCES Support_worker(ID_user)
);
```

Код створення процедур

```
USE RepairServiceDB;
GO

-- Процедура для додавання нового клієнта
CREATE PROCEDURE AddClient
    @FullName VARCHAR(100),
    @Phone VARCHAR(20),
    @Email VARCHAR(50),
    @Address VARCHAR(MAX)
AS
BEGIN
    DECLARE @UserID UNIQUEIDENTIFIER = NEWID();

    INSERT INTO [User] (ID_user, Full_name, Phone_number, Email) VALUES (@UserID, @FullName, @Phone, @Email);
    INSERT INTO Client (ID_user, Address) VALUES (@UserID, @Address);
END;
GO
```

```
-- Процедура для додавання нового працівника
CREATE PROCEDURE AddWorker
   @FullName VARCHAR(100),
   @Phone VARCHAR(20),
   @Email VARCHAR(MAX),
   @Position NVARCHAR(255),
   @OfficeID UNIQUEIDENTIFIER
AS
BEGIN
   DECLARE @UserID UNIQUEIDENTIFIER = NEWID();
   INSERT INTO [User] (ID_user, Full_name, Phone_number, Email) VALUES (@UserID, @FullName, @Phone, @Email);
   INSERT INTO Support_worker (ID_user, Position, ID_office) VALUES (@UserID, @Position, @OfficeID);
END;
GO

-- Процедура для подання заявки на консультацію-ремонт
CREATE PROCEDURE AddRepairRequest
   @ClientID UNIQUEIDENTIFIER,
   @RequestDate DATETIME,
   @VisitAddress VARCHAR(MAX),
   @RepairTypeID UNIQUEIDENTIFIER,
   @ProblemDescription TEXT,
   @Status VARCHAR(15)
AS
BEGIN
   INSERT INTO Repair_request (ID_request, Request_date, Visit_address, ID_user, ID_repair_type, Problem_description, Status)
   VALUES (NEWID(), @RequestDate, @VisitAddress, @ClientID, @RepairTypeID, @ProblemDescription, @Status);
END;
GO
```

```
-- Процедура для призначення ремонтного візиту
CREATE PROCEDURE ScheduleRepairVisit
    @WorkerID UNIQUEIDENTIFIER,
    @VisitDate DATETIME,
    @WorkDescription TEXT,
    @RequestID UNIQUEIDENTIFIER
AS
BEGIN
    INSERT INTO Repair_visit (ID_visit, ID_user, Visit_date, Work_description, ID_request)
    VALUES (NEWID(), @WorkerID, @VisitDate, @WorkDescription, @RequestID);
END;
GO
-- Процедура для завершення ремонту
CREATE PROCEDURE CompleteRepair
    @RequestID UNIQUEIDENTIFIER,
    @WorkerID UNIQUEIDENTIFIER,
    @CompletionDate DATE,
    @WorkDescription TEXT,
    @TotalCost DECIMAL(10,2)
AS
BEGIN
    INSERT INTO Completed_work (ID_request, ID_user, Completion_date, Work_description,
Total_cost)
    VALUES (@RequestID, @WorkerID, @CompletionDate, @WorkDescription, @TotalCost);
END;
GO
DROP PROCEDURE IF EXISTS [dbo].[GetClientMonthlyVisitStats];
GO
CREATE PROCEDURE [dbo].[GetClientMonthlyVisitStats]
    @ClientID UNIQUEIDENTIFIER,
    @Year INT,
    @Month INT
AS
```

```
BEGIN
   SELECT
      SUM(CASE WHEN Status = 'проведено' THEN 1 ELSE 0 END) AS CompletedRequests,
      SUM(CASE WHEN Status = 'скасовано' THEN 1 ELSE 0 END) AS CanceledRequests
   FROM Repair_request
   WHERE ID_user = @ClientID
     AND YEAR(Request_date) = @Year
     AND MONTH(Request_date) = @Month;
END;
GO
DROP PROCEDURE IF EXISTS [dbo].[GetWorkerPerformanceReport];
GO
CREATE PROCEDURE [dbo].[GetWorkerPerformanceReport]
AS
BEGIN
   SELECT
      u.ID_user AS WorkerID,
      u.Full_name AS WorkerName,
      COUNT(cw.ID_request) AS CompletedWorks,
      COALESCE(SUM(cw.Total_cost), 0) AS TotalEarned
   FROM [User] u
   INNER JOIN Support_worker sw ON u.ID_user = sw.ID_user
   LEFT JOIN Completed_work cw ON u.ID_user = cw.ID_user
   GROUP BY u.ID_user, u.Full_name;
END;
GO
DROP PROCEDURE IF EXISTS [dbo].[GetClientRepairReport];
GO
CREATE PROCEDURE GetClientRepairReport
AS
BEGIN
   SELECT
   u.ID_user,
```

```
    u.Full_name,
    COUNT(cw.ID_request) AS Completed_work_count,
    SUM(cw.Total_cost) AS Total_amount
        FROM Completed_work cw
        JOIN Repair_request rr ON cw.ID_request = rr.ID_request
        JOIN Client c ON rr.ID_user = c.ID_user
        JOIN [User] u ON c.ID_user = u.ID_user
        GROUP BY u.ID_user, u.Full_name;
END;
```

Код створення тригерів

```
USE RepairServiceDB;
GO
-- Дроп тригера перед створенням
IF OBJECT_ID('trg_CheckWorkerOffice', 'TR') IS NOT NULL
    DROP TRIGGER trg_CheckWorkerOffice;
GO
-- Створення тригера для перевірки офісу працівника
CREATE TRIGGER trg_CheckWorkerOffice
ON Support_worker
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @OfficeID UNIQUEIDENTIFIER;
    -- Отримання ID офісу з таблиці після вставки чи оновлення
    SELECT @OfficeID = ID_office FROM INSERTED;
-- Перевірка існування офісу
    IF NOT EXISTS (SELECT 1 FROM Office WHERE ID_office = @OfficeID)
    BEGIN
        PRINT 'Помилка: не знайдено вказаний офіс працівника.';
    END
END;
GO
```

```
-- Дроп тригера перед створенням
IF OBJECT_ID('trg_DeleteUserOnClientDelete', 'TR') IS NOT NULL
   DROP TRIGGER trg_DeleteUserOnClientDelete;
GO
-- Створення тригера для видалення користувача при видаленні клієнта
CREATE TRIGGER trg_DeleteUserOnClientDelete
ON Client
AFTER DELETE
AS
BEGIN
   DECLARE @UserID UNIQUEIDENTIFIER;
   SELECT @UserID = ID_user FROM DELETED;
   DELETE FROM [User]
   WHERE ID_user = @UserID;
END;
GO
-- Дроп тригера перед створенням
IF OBJECT_ID('trg_DeleteUserOnSupportWorkerDelete', 'TR') IS NOT NULL
   DROP TRIGGER trg_DeleteUserOnSupportWorkerDelete;
GO
-- Створення тригера для видалення користувача при видаленні працівника
CREATE TRIGGER trg_DeleteUserOnSupportWorkerDelete
ON Support_worker
AFTER DELETE
AS
BEGIN
   DECLARE @UserID UNIQUEIDENTIFIER;
   SELECT @UserID = ID_user FROM DELETED;
   DELETE FROM [User]
   WHERE ID_user = @UserID;
END;
GO
```

```
-- Дроп тригера перед створенням
IF OBJECT_ID('trg_PreventMeetingDeletion', 'TR') IS NOT NULL
   DROP TRIGGER trg_PreventMeetingDeletion;
GO

CREATE TRIGGER PreventRepairRequestDeletion
ON Repair_request
INSTEAD OF DELETE
AS
BEGIN
   IF EXISTS (SELECT 1 FROM deleted WHERE [Status] = 'проведено')
   BEGIN
      RAISERROR ('Неможливо видалити заявку на ремонт зі статусом "проведено".', 16, 1);
      RETURN;
   END
   ELSE
   BEGIN
      DELETE FROM Repair_request WHERE ID_request IN (SELECT ID_request FROM
deleted);
   END
END;
```

Код створення ролей, прав доступу, користувачів

```
USE RepairServiceDB;
GO
-- Видалення таблиці, якщо існує
IF OBJECT_ID('autification', 'U') IS NOT NULL
   DROP TABLE autification;
GO
-- Створення таблиці
CREATE TABLE autification (
   id_user INT IDENTITY NOT NULL,
```

```
    login VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,
    role VARCHAR(20) NOT NULL
);
GO
-- Додавання користувачів (логіном є номер телефону)
INSERT INTO autification (login, password, role) VALUES
('0964847364', '1234qwer', 'client'),
('0964534284', 'pfb3s043', 'client'),
('0954647878', 'oledx095', 'client'),
('0934567878', 'andr093', 'client'),
('0934567879', 'rdf0fsw3', 'client'),
('0684675342', 'dsg32224', 'support_worker'),
('0633212211', 'susdf002', 'support_worker'),
('0971224567', 'ssdfd003', 'support_worker'),
('0997643211', 'susg4s04', 'support_worker'),
('0501234567', 'd&j7wgs5', 'support_worker');
GO

-- Створення ролей
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'ClientRole')
    CREATE ROLE ClientRole;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'SupportWorkerRole')
    CREATE ROLE SupportWorkerRole;
GO

-- Призначення прав ролям
GRANT SELECT, INSERT, UPDATE ON SCHEMA :: dbo TO ClientRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA :: dbo TO SupportWorkerRole;
GO
```

```
-- Створення логінів, користувачів та додавання до ролей
DECLARE @login VARCHAR(50), @password VARCHAR(50), @role VARCHAR(20), @sql NVARCHAR(MAX);
DECLARE user_cursor CURSOR FOR
   SELECT login, password, role FROM autification;
OPEN user_cursor;
FETCH NEXT FROM user_cursor INTO @login, @password, @role;
WHILE @@FETCH_STATUS = 0
BEGIN
   -- Видалити логін, якщо такий існує
   IF EXISTS (SELECT * FROM sys.server_principals WHERE name = @login)
   BEGIN
      SET @sql = N'DROP LOGIN [' + @login + '];';
      EXEC (@sql);
   END
   -- Створення логіна
   SET @sql = N'CREATE LOGIN [' + @login + '] WITH PASSWORD = ''' + @password + ''';';
   EXEC (@sql);
   -- Створення користувача в БД
   SET @sql = N'CREATE USER [' + @login + '] FOR LOGIN [' + @login + '];';
   EXEC (@sql);
   -- Призначення ролі користувачу
   IF @role = 'client'
      EXEC sp_addrolemember 'ClientRole', @login;
   ELSE IF @role = 'support_worker'
      EXEC sp_addrolemember 'SupportWorkerRole', @login;
   FETCH NEXT FROM user_cursor INTO @login, @password, @role;
END
CLOSE user_cursor;
DEALLOCATE user_cursor;
```

ДОДАТОК Д

Код методу додавання заявки на візит

Сторінок 3

Київ-2025

Код методу додавання заявки на візит

```
private void Add_meeting_Click(object sender, EventArgs e)
    {
        // Об'єднання дати та часу
        DateTime selectedDate = dateTimePicker4.Value.Date;
        TimeSpan selectedTime = dateTimePicker3.Value.TimeOfDay;
        DateTime requestDateTime = selectedDate + selectedTime;
        string address = richTextBox4.Text.Trim();
        string problemDescription = richTextBox5.Text.Trim();
        if (comboBox2.SelectedValue == null)
        {
            MessageBox.Show("Будь ласка, оберіть тип ремонту.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        Guid repairTypeId = (Guid)comboBox2.SelectedValue;
        // Перевірка на дублювання заявки на той самий час
        string duplicateCheckQuery = @"
        SELECT COUNT(*) FROM Repair_request
        WHERE ID_user = @UserId AND Request_date = @RequestDate";
        using (SqlCommand duplicateCheckCmd = new SqlCommand(duplicateCheckQuery, connection))
        {
            duplicateCheckCmd.Parameters.AddWithValue("@UserId", clientID);
            duplicateCheckCmd.Parameters.AddWithValue("@RequestDate", requestDateTime);
            int existingCount = (int)duplicateCheckCmd.ExecuteScalar();
            if (existingCount > 0)
            {
MessageBox.Show("У вас вже є заявка на цей час. Будь ласка, оберіть інший час.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
        }
    }
    // Перевірка на інтервал не менше 2 годин між заявками
    string intervalCheckQuery = @"
    SELECT COUNT(*) FROM Repair_request
    WHERE ID_user = @UserId AND ABS(DATEDIFF(MINUTE, Request_date, @RequestDate)) < 120";
    using (SqlCommand intervalCheckCmd = new SqlCommand(intervalCheckQuery, connection))
    {
        intervalCheckCmd.Parameters.AddWithValue("@UserId", clientID);
        intervalCheckCmd.Parameters.AddWithValue("@RequestDate", requestDateTime);
        int intervalCount = (int)intervalCheckCmd.ExecuteScalar();
        if (intervalCount > 0)
        {
            MessageBox.Show("Між заявками має бути інтервал не менше 2 годин. Будь ласка, оберіть інший час.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }
    // Додавання нової заявки
    string insertQuery = @"
    INSERT INTO Repair_request (Request_date, Visit_address, ID_user, ID_repair_type, Problem_description, Status)
    VALUES (@RequestDate, @VisitAddress, @UserId, @RepairTypeId, @ProblemDescription, 'призначено')";
    using (SqlCommand insertCmd = new SqlCommand(insertQuery, connection))
    {
        insertCmd.Parameters.AddWithValue("@RequestDate", requestDateTime);
        insertCmd.Parameters.AddWithValue("@VisitAddress", address);
        insertCmd.Parameters.AddWithValue("@UserId", clientID);
```

```
            insertCmd.Parameters.AddWithValue("@RepairTypeId", repairTypeId);
            insertCmd.Parameters.AddWithValue("@ProblemDescription", problemDescription);
            int rowsAffected = insertCmd.ExecuteNonQuery();
            if (rowsAffected > 0)
            {
                MessageBox.Show("Заявку успішно додано!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
                LoadMeetingsForClient();
                LoadScheduledMeetings();
            }
            else
            {
                MessageBox.Show("Не вдалося додати заявку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
```

ДОДАТОК Е

Грамота Всеукраїнської студентської олімпіади 2024-2025 н.р. зі спеціальності «Комп’ютерні науки»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ГРАМОТА

НАГОРОДЖУЄТЬСЯ

Ломянський Богдан

студент(ка)

факультету (ННІ) Інформаційних технологій

за здобуте II місце

у І етапі Всеукраїнської студентської олімпіади 2024-2025 н.р.

зі спеціальності Комп'ютерні науку

Ректор Вадим ТКАЧУК