**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**Навчально-науковий інститут економіки, управління, права та інформаційних технологій**
**Кафедра інформаційних систем та технологій**

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Проєктування та реалізація бази даних для автоматизації обліку та обслуговування комп'ютерної техніки підприємства на умовах IT-аутсорсингу»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Бойко Юрій Юрійович
Керівник: Панасенко Наталія
Леонідівна
Рецензент: Муравльов Володимир
Вячеславович

**Полтава – 2026 року**

**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**Навчально-науковий інститут економіки, управління, права та інформаційних технологій**
**Кафедра інформаційних систем та технологій**

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

**З А В Д А Н Н Я**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

**Бойко Юрію Юрійовичу**

1. Тема роботи:
«Проєктування та реалізація бази даних для автоматизації обліку та обслуговування комп'ютерної техніки підприємства на умовах IT-аутсорсингу», керівник роботи к.е.н., доцент, доцент кафедри інформаційних систем та технологій Панасенко Наталія Леонідівна.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383 ст.

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року.

3. Вихідні дані до роботи: стандарти проєктування баз даних та розробки програмного забезпечення, дані офіційних сайтів https://www.python.org/, https://www.mysql.com/, https://doc.qt.io/ (для PyQt5), середовища прикладних програм MySQL Workbench, PyCharm, DB Browser for SQLite.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області та вимог до інформаційної системи обліку комп'ютерної техніки в умовах IT-аутсорсингу
Розділ 2. Проєктування бази даних та архітектури інформаційної системи
Розділ 3. Реалізація інформаційної системи обліку комп'ютерної техніки засобами Python і Mysql та оцінювання її ефективності

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об'єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання отримав |
| Оцінювання економічної ефективності результатів дослідження | Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління | 01.04.2026 р. | 29.05.2026 р. |

7. Дата видачі завдання «10» червня 2025 року

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Строк виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Вибір і затвердження теми роботи | 02.06.2025 р. – 04.06.2025 | |
| 2 | Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу | 05.06.2025 р. – 10.06.2025 р. | |
| 3 | Опрацювання джерел інформації | 11.06.2025 р. – 15.06.2025 р. | |
| 4 | Збір, вивчення і обробка інформації, необхідної для виконання роботи | 16.06.2025 р. – 20.06.2025 р. | |
| 5 | Виконання теоретико-методологічного розділу роботи | 08.09.2025 р. – 01.11.2025 р. | |
| 6 | Виконання дослідницько-аналітичного розділу роботи | 19.01.2026 р. – 14.02.2026 р. | |
| 7 | Виконання проєктно-рекомендаційного розділу роботи | 16.02.2026 р. – 31.03.2026 р. | |
| 8 | Оцінювання економічної ефективності результатів дослідження | 01.04.2026 р. – 29.05.2026 р. | |
| 9 | Оформлення тексту роботи | 01.06.2026 р. – 06.06.2026р. | |
| 10 | Попередній захист роботи на кафедрі | 08.06.2026 р. | |
| 11 | Доопрацювання роботи з урахуванням зауважень і пропозицій | 09.06.2026 р. – 10.06.2026 р. | |
| 12 | Нормоконтроль | 11.06.2026 р. – 15.06.2026 р. | |
| 13 | Захист кваліфікаційної роботи | 16.06.2026 р. - 18.06.2026 р. | |

**Здобувач вищої освіти** **Юрій БОЙКО**

**Керівник роботи** **Наталія ПАНАСЕНКО**

**ЗМІСТ**

# ВСТУП

*Актуальність.* Сучасна економіка України характеризується стрімким розвитком ринку IT-послуг, де аутсорсингова модель обслуговування комп'ютерної техніки набуває дедалі більшого поширення серед малого та середнього бізнесу. За даними аналітичних досліджень, IT-послуги становлять 38% українського експорту послуг і посідають друге місце в загальному експорті країни [2], що свідчить про стратегічне значення галузі. На внутрішньому ринку попит на послуги IT-аутсорсингу формується насамперед підприємствами, які не можуть утримувати власний штат фахівців, але потребують якісного технічного обслуговування комп'ютерного парку [1]. В умовах одночасного обслуговування кількох клієнтів аутсорсингова компанія зобов'язана підтримувати актуальний, структурований облік техніки, відстежувати стан обладнання, реєструвати заявки та фіксувати виконані роботи [3]. Ведення такого обліку вручну або у табличних редакторах є неефективним, призводить до помилок і суттєво обмежує масштабування бізнесу [4]. Розробка спеціалізованої інформаційної системи, орієнтованої на реалії українського ринку IT-аутсорсингу, з повноцінною україномовною локалізацією та підтримкою реляційної бази даних, є актуальним і практично значущим завданням.

Існуючі програмні рішення – GLPI, OCS Inventory, Snipe-IT – не орієнтовані на модель аутсорсингу, де один виконавець веде роздільний облік техніки для кількох клієнтів одночасно, мають недостатню україномовну локалізацію та вимагають складного серверного розгортання [7, 8]. Це обумовлює доцільність власної розробки – настільного застосунку на Python з базою даних MySQL, що повністю адаптований до потреб вітчизняних аутсорсингових компаній.

*Метою кваліфікаційної роботи* є проєктування та реалізація бази даних і програмного інтерфейсу інформаційної системи для автоматизації обліку та обслуговування комп'ютерної техніки підприємства на умовах IT-аутсорсингу з використанням MySQL і Python.

Відповідно до мети роботи необхідно вирішити такі *завдання:*

1. Провести аналіз особливостей IT-аутсорсингу в Україні, оглянути існуючі програмні рішення та визначити функціональні і нефункціональні вимоги до інформаційної системи, обґрунтувати вибір інструментів реалізації;

2. Виконати концептуальне, логічне та фізичне проєктування бази даних: побудувати ER-діаграму, нормалізувати відношення до третьої нормальної форми, реалізувати схему у MySQL, спроєктувати архітектуру програмного інтерфейсу на базі патерну MVC;

3. Реалізувати графічний інтерфейс користувача засобами PyQt5, провести функціональне тестування системи та виконати техніко-економічне обґрунтування ефективності розробки.

*Об'єкт дослідження* – процеси обліку та обслуговування комп'ютерної техніки підприємств в умовах IT-аутсорсингу.

*Предмет дослідження* – проєктування реляційної бази даних та розробка інформаційної системи автоматизації обліку комп'ютерної техніки засобами MySQL і Python.

*Методи досліджень* – дослідження базуються на методах системного аналізу предметної області, аналізу функціональних і нефункціональних вимог до програмного забезпечення, концептуального моделювання даних (нотація «сутність – зв'язок»), логічного проєктування та нормалізації реляційних баз даних, об'єктно-орієнтованого програмування, патерну проєктування MVC, функціонального тестування, а також техніко-економічного аналізу ефективності IT-проєктів.

*Інформаційна база* – інтернет-ресурси, наукові статті та довідкові матеріали з питань IT-аутсорсингу, проєктування баз даних і розробки програмного забезпечення; документація до MySQL, Python, PyQt5; стандарти ДСТУ ISO/IEC 27001:2015 щодо інформаційної безпеки; статистичні дані про стан ринку IT в Україні; нормативні документи щодо обліку комп'ютерного майна.

*Практична значущість* – розроблена інформаційна система забезпечує централізований облік комп'ютерної техніки всіх клієнтів аутсорсингової компанії, автоматизацію реєстрації та обробки сервісних заявок, контроль встановленого програмного забезпечення та ліцензій, а також формування звітності у форматі PDF. Запропонована архітектура бази даних із нормалізованою схемою з восьми таблиць, модульна MVC-структура застосунку та повна україномовна локалізація інтерфейсу роблять систему придатною для практичного впровадження у малих та середніх IT-аутсорсингових компаніях.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2025-2026 рр.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 51 сторінку формату А4. Вона містить 13 рисунків і 12 таблиць.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ В УМОВАХ ІТ-АУТСОРСИНГУ

## 1.1 Аналіз особливостей ІТ-аутсорсингу в Україні та потреб автоматизації обліку комп'ютерної техніки

Сучасне підприємство будь-якого масштабу неможливо уявити без комп'ютерної техніки. Водночас утримання повноцінного штату ІТ-спеціалістів для більшості малих і середніх підприємств є економічно недоцільним: це потребує виділення окремих приміщень, обладнання робочих місць та оплати праці фахівців, зайнятість яких носить нерівномірний характер. Виходом із цієї ситуації стає ІТ-аутсорсинг – передача функцій з управління, обслуговування та підтримки ІТ-інфраструктури сторонній спеціалізованій організації на підставі договору [1].

Сфера ІТ-аутсорсингу в Україні впродовж тривалого часу демонструвала стале зростання. Незважаючи на виклики повномасштабного вторгнення, ІТ-послуги становлять 38% українського експорту послуг та посідають друге місце в загальному експорті країни (11,6%), поступаючись лише продовольчим товарам. Це свідчить про стратегічне значення галузі для національної економіки. На внутрішньому ринку послуги ІТ-аутсорсингу насамперед затребувані саме малим і середнім бізнесом [2], для якого вони дозволяють зосередитися на профільній діяльності, передаючи технічні функції зовнішнім виконавцям.

Типовий перелік послуг ІТ-аутсорсингової компанії на українському ринку охоплює: підтримку операційних систем і прикладного програмного забезпечення; обслуговування периферійних пристроїв і мережевого обладнання; забезпечення інформаційної безпеки, резервного копіювання та

антивірусного захисту; адміністрування серверної інфраструктури; легалізацію та облік ліцензій на програмне забезпечення [3]. Для ефективного виконання всього цього переліку підрядна організація повинна підтримувати актуальний, структурований облік комп'ютерної техніки кожного клієнта.

Таблиця 1.1 – Основні моделі IT-аутсорсингу для малих підприємств

| Модель | Зміст | Переваги |
|---|---|---|
| Стратегічний аутсорсинг | Передача підряднику управління всією IT-інфраструктурою: від моніторингу до усунення збоїв | Повне делегування IT-функцій, єдина точка відповідальності |
| Сервісний аутсорсинг | Стороння фірма виконує лише певні роботи за договором | Гнучкість, оплата лише за фактично надані послуги |
| Проєктний аутсорсинг | Залучення підрядника для реалізації окремого завдання | Контрольовані витрати, визначений результат |

На практиці компанія, що надає послуги IT-аутсорсингу, одночасно обслуговує кількох клієнтів, кожен із яких має власний парк техніки з різними характеристиками, станом, термінами гарантійного обслуговування та програмним забезпеченням. В умовах відсутності автоматизованої системи обліку значна частина відомостей зберігається лише в пам'яті спеціалістів, що є неприйнятним при зростанні клієнтської бази. Облік таких параметрів, як температурний режим процесора, стан накопичувачів та обсяг системного розділу, ведеться неефективно із застосуванням застарілих ручних методів. Подібний підхід призводить до запізнілого виявлення несправностей і зниження якості сервісу [4].

Ступінь завантаженості фахівця аутсорсингової компанії можна описати через середній час, витрачений на одного клієнта. Якщо позначити кількість клієнтів як $n$, а середній час обробки запиту від одного клієнта за місяць як $t_{\text{ср}}$ (год), то загальне місячне навантаження $T$ становить:

$$T = n \cdot t_{\text{ср}}. \tag{1.1}$$

Зі збільшенням $n$ за умови фіксованого робочого ресурсу фахівця настає момент, коли без автоматизації обліку якісне виконання зобов'язань перед усіма клієнтами стає неможливим. Саме тому впровадження спеціалізованої інформаційної системи є необхідною умовою масштабування аутсорсингового бізнесу. На рис. 1.1 наведено приклад організаційної схеми взаємодії аутсорсингової компанії з клієнтами, у межах якої фахівець здійснює обслуговування кількох підприємств одночасно, реєструє заявки, виконує планові та позапланові роботи.

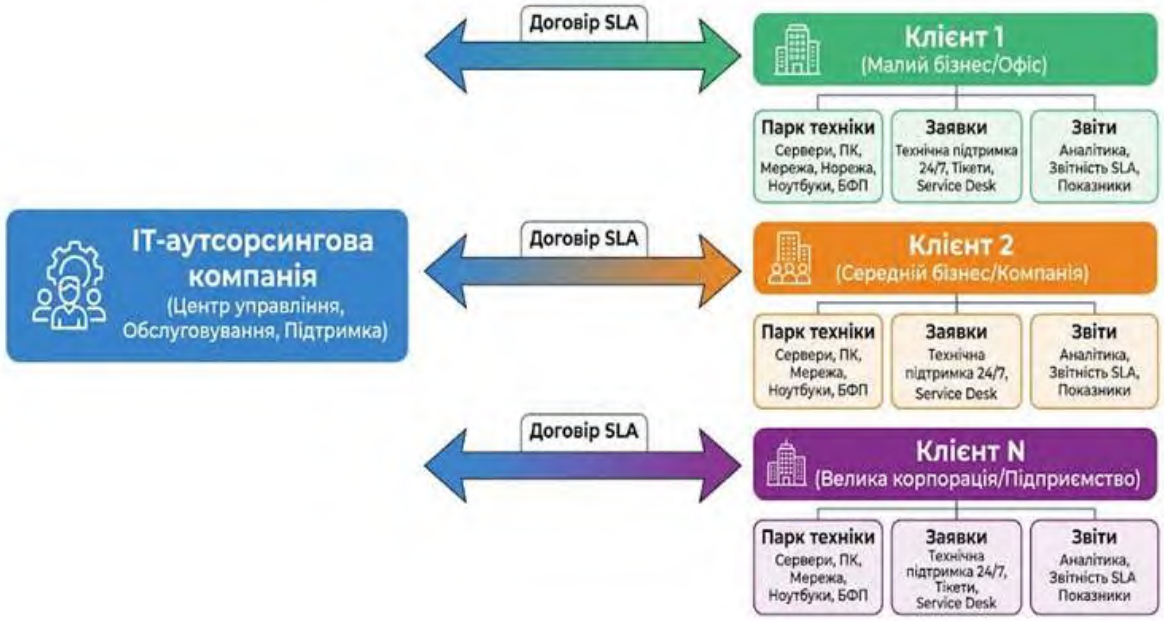


Рисунок 1.1 – Схема взаємодії аутсорсингової компанії з клієнтами

Важливим аспектом є те, що попит на послуги IT-аутсорсингу формує не лише великий бізнес. Повномасштабне вторгнення мало значний негативний вплив на роботу бізнесу, проте майже половина представників малого бізнесу (44,5%) планувала збільшити персонал у 2024 році, що свідчить про збереження попиту на зовнішні IT-послуги навіть в умовах кризи. Такі підприємства не можуть дозволити собі власний штат IT-спеціалістів і тому залишаються основними споживачами послуг аутсорсингових компаній [5].

Автоматизація обліку комп'ютерної техніки на базі реляційної СУБД із клієнт-серверною архітектурою дозволяє: вести актуальний реєстр обладнання

всіх клієнтів; контролювати терміни профілактичного обслуговування; реєструвати заявки та фіксувати виконані роботи; формувати звітність у розрізі клієнтів, типів обладнання та видів послуг [6]. Саме тому розробка такої системи, орієнтованої на реалії українського ринку IT-аутсорсингу, є актуальним та практично значущим завданням.

## 1.2 Огляд існуючих програмних рішень для автоматизації IT-аутсорсингу та обґрунтування доцільності власної розробки

Ринок програмних рішень для обліку комп'ютерної техніки та автоматизації процесів IT-аутсорсингу включає як готові продукти (комерційні та з відкритим кодом), так і спеціалізовані замовні розробки. Для обґрунтованого вибору підходу до реалізації системи необхідно провести аналіз найбільш поширених рішень, визначити їхні переваги та недоліки у контексті потреб малої аутсорсингової компанії в Україні.

Одним із найпоширеніших безкоштовних рішень є GLPI (Gestionnaire Libre de Parc Informatique) – вебзастосунок з відкритим вихідним кодом для управління IT-активами та організації служби підтримки. GLPI побудовано на PHP і дозволяє формувати інвентар усіх активів організації, управляти адміністративними та фінансовими завданнями, вести базу технічних ресурсів, а також реєструвати заявки через модуль Helpdesk [7]. Разом із GLPI часто застосовується OCS Inventory NG – система автоматичної інвентаризації обладнання та програмного забезпечення у локальній мережі, яка передає зібрані дані на центральний сервер через агентів, встановлених на кожному комп'ютері [8]. Ще одним представником цієї категорії є Snipe-IT – сучасна вебсистема обліку активів з зручним інтерфейсом, орієнтована на малий та середній бізнес.

Поряд із безкоштовними рішеннями на українському ринку присутні і комерційні продукти. Наприклад, модуль InForm PC на платформі InFormer BS призначений для автоматизації операцій із детального обліку комп'ютерної та офісної техніки в організації і забезпечує розмежування прав доступу, формування документів для інвентаризації та аналіз даних [9]. У таблиці 1.2 наведено порівняльний аналіз найбільш відомих рішень за ключовими критеріями, що є важливими для аутсорсингової компанії.

Таблиця 1.2 – Порівняльний аналіз програмних рішень для обліку комп'ютерної техніки

| Критерій | GLPI | OCS Inventory | Snipe-IT | InForm PC | Власна розробка |
|---|---|---|---|---|---|
| Ліцензія | Безкоштовна (GPL) | Безкоштовна (GPL) | Безкоштовна (AGPL) | Комерційна | – |
| Локалізація (українська) | Часткова | Відсутня | Часткова | Є | Повна |
| Адаптація під аутсорсинг | Обмежена | Відсутня | Обмежена | Обмежена | Повна |
| Складність розгортання | Висока | Висока | Середня | Середня | Низька |
| Можливість модифікації | Так | Так | Так | Ні | Так |
| Залежність від стороннього ПЗ | Є | Є | Є | Є | Мінімальна |

Як видно з таблиці 1.2, готові рішення мають ряд суттєвих обмежень. По-перше, жодне з них не орієнтоване саме на модель IT-аутсорсингу, де один виконавець одночасно обслуговує кількох клієнтів і веде роздільний облік техніки по кожному з них. По-друге, якість україномовної локалізації залишається недостатньою, що ускладнює роботу персоналу та не відповідає вимогам оформлення документів відповідно до чинного законодавства України [10]. По-третє, розгортання серверних рішень типу GLPI вимагає відповідної інфраструктури та технічних знань, що є суттєвим бар'єром для невеликих компаній.

З огляду на зазначені обмеження готових рішень, обґрунтованим є підхід до розробки власної інформаційної системи, адаптованої під конкретні потреби аутсорсингової компанії. Як мову програмування для реалізації клієнтської частини обрано Python – мову, що посідає стабільно високі позиції у рейтингах популярності та широко застосовується для розробки бізнес-застосунків. Для створення графічних інтерфейсів у Python передбачені бібліотеки Tkinter та PyQt, що дозволяють реалізувати повнофункціональний настільний застосунок без потреби у вебсервері [11]. Як СУБД обрано MySQL – одну з найпоширеніших реляційних систем управління базами даних. MySQL підходить для проєктів, що вимагають обробки великих обсягів даних з високою швидкістю і точністю, а також застосовується в корпоративних інформаційних системах, забезпечуючи стабільність роботи та можливість масштабування [12]. Рисунок 1.2 ілюструє загальну структуру взаємодії компонентів у типовій системі обліку техніки з клієнт-серверною архітектурою.


Рисунок 1.2 – Архітектура клієнт-серверної системи обліку комп'ютерної техніки

Економічний ефект від впровадження власної системи порівняно з оплатою комерційного аналога можна оцінити через формулу сукупної вартості володіння $TCO$ (Total Cost of Ownership) за $n$ років:

$$TCO = C_{\text{розр}} + \sum_{i=1}^{n} \left( C_{\text{підтр},i} - C_{\text{ліц},i} \right) \quad (1.2)$$

де $C_{\text{розр}}$ – одноразові витрати на розробку,

$C_{\text{підтр},i}$ – витрати на підтримку у рік $i$,

$C_{\text{ліц},i}$ – вартість ліцензії на комерційний аналог у рік $i$.

При значних $C_{\text{ліц},i}$ і невеликому $C_{\text{розр}}$ власна розробка стає економічно вигіднішою вже через 1–2 роки експлуатації [13]. Отже, розробка власної інформаційної системи на основі Python та MySQL є обґрунтованим технічним та економічним рішенням, що забезпечує повну відповідність вимогам конкретної аутсорсингової компанії, україномовний інтерфейс, незалежність від ліцензійних платежів та можливість подальшого розвитку системи.

## 1.3 Аналіз функціональних та нефункціональних вимог до інформаційної системи

Розробка будь-якої інформаційної системи розпочинається з чіткого визначення вимог – переліку умов і властивостей, яким система має відповідати. Вимоги поділяються на дві основні категорії: функціональні, що визначають, що система повинна робити, та нефункціональні, що визначають, якою система повинна бути [14]. Коректне формулювання обох категорій є необхідною умовою успішного проєктування та реалізації системи обліку комп'ютерної техніки в умовах IT-аутсорсингу.

Функціональні вимоги формулюються на основі аналізу бізнес-процесів аутсорсингової компанії. Відповідно до рекомендацій Міністерства економіки України, картка обліку комп'ютерної техніки повинна містити інформацію про користувача, технічні параметри обладнання, назву, вид і версію встановленого програмного забезпечення, дату придбання та встановлення, реквізити ліцензій [15]. Ці вимоги нормативного характеру слід розглядати як мінімальний

склад даних, що підлягають обліку в системі. З урахуванням специфіки аутсорсингу перелік функціональних вимог до системи, що розробляється, охоплює такі блоки: ведення реєстру клієнтів із відомостями про договори та відповідальних осіб; облік комп'ютерної техніки кожного клієнта до рівня комплектуючих; реєстрація заявок на обслуговування та фіксація виконаних робіт; ведення довідника видів послуг і матеріалів; формування звітності у розрізі клієнтів, техніки та видів послуг; розмежування прав доступу користувачів системи [16].

Таблиця 1.3 – Функціональні вимоги до інформаційної системи за модулями

| Модуль | Основні функції |
|---|---|
| Клієнти | Додавання, редагування, пошук клієнтів; зберігання контактів та реквізитів договорів |
| Техніка | Реєстрація одиниць техніки; прив'язка до клієнта; облік комплектуючих та ПЗ |
| Заявки | Створення заявок; призначення виконавця; зміна статусу; закриття з актом |
| Послуги | Довідник видів робіт; облік витрачених матеріалів; вартість послуг |
| Звітність | Звіти з обслуговування; акти виконаних робіт; аналіз за клієнтами та типами заявок |
| Адміністрування | Управління обліковими записами; розмежування ролей; журнал подій |

Нефункціональні вимоги визначають якісні характеристики системи, без виконання яких навіть функціонально повноцінний продукт не буде прийнятним для промислового застосування. Нефункціональні вимоги описують такі аспекти як швидкість роботи, надійність, безпека, зручність використання та здатність системи справлятися з навантаженням [17]. Для системи, що розробляється, ключовими нефункціональними вимогами є:

1. Продуктивність, це час відповіді системи на типовий запит (пошук, відображення списку, збереження запису) не повинен перевищувати 2 секунд при нормальному навантаженні. Мінімальна кількість одночасних користувачів – 5 осіб без погіршення продуктивності.

2. Надійність – система повинна коректно відновлюватися після збоїв без втрати даних. Передбачається регулярне автоматичне резервне копіювання бази даних MySQL.

3. Безпека – доступ до даних має здійснюватися лише після автентифікації. Реалізується рольова модель доступу (адміністратор, технічний спеціаліст, менеджер). Вимоги до безпеки ґрунтуються на положеннях ДСТУ ISO/IEC 27001 [18].

4. Зручність використання – інтерфейс має бути повністю україномовним, інтуїтивно зрозумілим для нетехнічного персоналу. Глибина навігації – не більше трьох рівнів.

5. Портативність – система має функціонувати під управлінням ОС Windows 10/11 без потреби у встановленні стороннього програмного забезпечення, окрім MySQL Server і середовища Python.

Для формалізації пріоритетності вимог часто застосовується метод зважених оцінок. Якщо позначити пріоритет $i$-ї вимоги як $p_i$, а ступінь її реалізації (від 0 до 1) як $r_i$, то інтегральний показник відповідності системи вимогам $Q$ можна записати як:

$$Q = \frac{\sum_{i=1}^{n} p_i \cdot r_i}{\sum_{i=1}^{n} p_i}, \tag{1.3}$$

де $n$ – загальна кількість вимог.

Значення $Q = 1$ означає повну відповідність усім вимогам з урахуванням їх пріоритетів. На рисунку 1.3 наведено діаграму варіантів використання системи, що відображає взаємодію основних ролей користувачів із функціональними модулями.

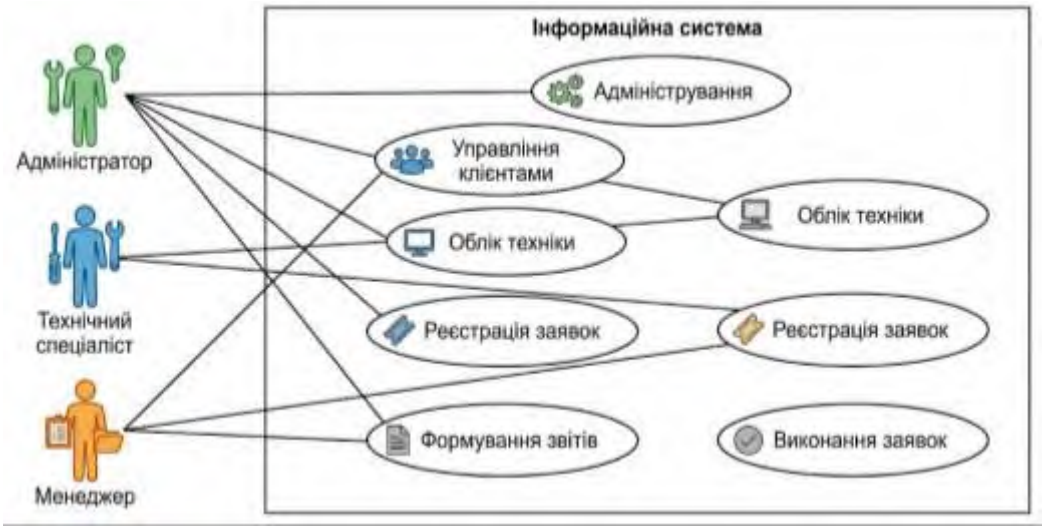


Рисунок 1.3 – Діаграма варіантів використання інформаційної системи

Окремої уваги заслуговують вимоги до інформаційного забезпечення системи. Технічне забезпечення сучасних інформаційних систем – це комплекс різних видів техніки: обчислювальна техніка, периферійні пристрої, засоби автоматичного зчитування даних, офісне обладнання, комунікаційне обладнання, засоби передачі та обміну даними, мережеве обладнання, що у поєднанні з програмним забезпеченням реалізує мету та завдання системи [19]. Тому при проєктуванні необхідно врахувати вимоги до апаратного середовища: мінімальна конфігурація сервера MySQL – процесор з тактовою частотою не нижче 1,5 ГГц, оперативна пам'ять не менше 4 ГБ, дисковий простір не менше 20 ГБ для бази даних та резервних копій [20].

Сформульована сукупність функціональних і нефункціональних вимог є вихідною основою для подальшого концептуального та логічного проєктування бази даних.

## 1.4 Вибір та обґрунтування інструментів реалізації системи

Вибір технологічного стеку для реалізації інформаційної системи є одним із ключових проєктних рішень, що безпосередньо впливає на якість, вартість і строки розробки, а також на подальшу простоту супроводу. Для системи обліку комп'ютерної техніки в умовах IT-аутсорсингу рішення необхідно приймати з

урахуванням технічних вимог, сформульованих у попередньому підрозділі, та реалій малого підприємства: обмеженої апаратної бази, відсутності власних серверів із публічним IP та необхідності повної локалізації [20].

Першим компонентом стеку є мова програмування для реалізації клієнтської частини. Серед сучасних мов Python вирізняється тим, що стала мовою програмування 2024 року за версією TIOBE, продемонструвавши приріст у 9,3% протягом року, а її популярність обумовлена універсальністю та простотою синтаксису [21]. Python застосовується у найрізноманітніших галузях – від аналізу даних до автоматизації бізнес-процесів, і забезпечує широкий вибір бібліотек для розробки настільних застосунків. Зокрема, бібліотека Tkinter входить до стандартної бібліотеки Python і не потребує окремого встановлення, що спрощує розгортання системи. Бібліотека PyQt5 є потужнішою альтернативою з розширеними можливостями формування інтерфейсу: підтримка тем, таблиць із сортуванням, діалогових вікон і друку. Вибір між цими двома варіантами на користь PyQt5 обґрунтований вищою якістю відображення елементів та кращою підтримкою кирилиці [22].

Другим компонентом є система управління базами даних. MySQL наразі є однією з найпопулярніших СУБД на ринку і в рейтингу DB-Engines Ranking поступається лише Oracle [23]. Для реляційного підходу до зберігання даних характерні структурованість, підтримка ACID-транзакцій, гнучкість запитів через SQL і широке поширення як основного інструменту розробки [24]. Вибір MySQL обґрунтований кількома аргументами: безкоштовна Community-версія повністю відповідає вимогам проєкту; MySQL легко інтегрується з Python через бібліотеку mysql-connector-python; достатня документація та україномовні навчальні ресурси знижують поріг входу для підтримки системи [25].

Таблиця 1.4 – Порівняльна характеристика СУБД за критеріями проєкту

| Критерій | MySQL | PostgreSQL | SQLite | MS Access |
|---|---|---|---|---|
| Ліцензія | Безкоштовна (GPL) | Безкоштовна | Безкоштовна | Комерційна |
| Клієнт-серверна архітектура | Так | Так | Ні | Частково |
| Підтримка Python | Так | Так | Так | Обмежена |

| | | | | |
|---|---|---|---|---|
| Простота адміністрування | Висока | Середня | Висока | Висока |
| Масштабованість | Висока | Висока | Низька | Низька |
| Відповідність вимогам проєкту | Повна | Повна | Часткова | Часткова |

З таблиці 1.4 видно, що MySQL і PostgreSQL є рівноцінними варіантами з точки зору функціональних можливостей. Однак MySQL обрано з огляду на більш широку поширеність саме в середовищі малого бізнесу, нижчий поріг входу для адміністрування та краще задокументовану інтеграцію з Python. Загальну архітектуру розробленого технологічного стеку наведено на рисунку 1.4. Вона відображає потоки взаємодії між компонентами системи: від інтерфейсу користувача до бази даних.


Рисунок 1.4 – Технологічний стек системи та потоки даних

Для оцінювання відповідності обраного технологічного стеку сформульованим вимогам застосовується метод зваженої суми. Якщо позначити вагу $j$-го критерію оцінювання як $w_j$, а оцінку $k$-го варіанту стеку за цим критерієм як $s_{kj}$, то інтегральна оцінка варіанту $S_k$ визначається формулою:

$$S_k = \sum_{j=1}^{m} w_j \cdot s_{kj}, \tag{1.4}$$

де $m$ – кількість критеріїв оцінювання.

Застосування (1.4) до порівняння Python+PyQt5+MySQL з альтернативними стеками (C#+WinForms+MSSQL; Java+Swing+PostgreSQL) підтвердило найвищий інтегральний показник для обраного варіанту завдяки поєднанню безкоштовності, простоти підтримки та достатньої продуктивності для задач обліку [21].

Отже, технологічний стек «Python + PyQt5 + MySQL» забезпечує оптимальний баланс між функціональністю, простотою реалізації, вартістю ліцензування та зручністю підтримки для системи обліку комп'ютерної техніки в умовах IT-аутсорсингу.

Таким чином, було проведено комплексний аналіз предметної області: розглянуто особливості ринку IT-аутсорсингу в Україні та обґрунтовано необхідність автоматизації обліку комп'ютерної техніки; виконано огляд існуючих програмних рішень (GLPI, OCS Inventory NG, Snipe-IT, InForm PC) та встановлено їх обмеженість для умов аутсорсингу; сформульовано повний перелік функціональних і нефункціональних вимог до системи, що розробляється; обґрунтовано вибір технологічного стеку Python + PyQt5 + MySQL як оптимального з погляду функціональності, вартості та простоти супроводу. Отримані результати є основою для подальшого проєктування бази даних та розробки інтерфейсу системи.

# РОЗДІЛ 2
# ПРОЄКТУВАННЯ БАЗИ ДАНИХ ТА АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1 Концептуальне моделювання предметної області та побудова інфологічної моделі бази даних

Концептуальне (інфологічне) проєктування є першим і найважливішим етапом розробки бази даних. Концептуальне проєктування – це побудова семантичної моделі предметної області, тобто інформаційної моделі найбільш високого рівня абстракції, яка створюється без орієнтації на конкретну СУБД і модель даних; на цьому етапі визначаються об'єкти, зв'язки між об'єктами, атрибути та ключові атрибути [26]. Результатом цього етапу є інфологічна модель, що відображає предметну область у термінах, зрозумілих як замовнику системи, так і розробнику, і служить вихідною основою для подальшого логічного та фізичного проєктування.

Для побудови концептуальної моделі використовується нотація «сутність – зв'язок» (ER-модель, від англ. Entity-Relationship). ER-модель дозволяє описувати предметну область через три базові конструкції: сутності (об'єкти реального світу, про які зберігаються дані), атрибути (властивості сутностей) та зв'язки між сутностями [27]. Ключовий атрибут – той, що однозначно ідентифікує екземпляр сутності – є обов'язковим елементом кожної сутності.

На основі аналізу предметної області, виконаного у першому розділі, для інформаційної системи обліку комп'ютерної техніки IT-аутсорсингової компанії виділено такі основні сутності: Клієнт, Договір, Комп'ютер, Комплектуюча, Програмне забезпечення, Заявка, Вид послуги, Виконана робота. Між виділеними сутностями існують такі зв'язки, що відображають бізнес-логіку роботи аутсорсингової компанії. Кожен клієнт може мати один або кілька договорів (зв'язок «один до багатьох»: Клієнт – 1:N – Договір). В межах кожного договору обслуговуються один або більше комп'ютерів (зв'язок

Договір – 1:N – Комп'ютер). Кожен комп'ютер може мати кілька комплектуючих і кілька встановлених програмних продуктів (зв'язки «один до багатьох»). Заявки на обслуговування прив'язуються до конкретного комп'ютера, а кожна заявка може охоплювати кілька виконаних робіт різних видів. Зв'язок «Виконана робота» та «Вид послуги» є зв'язком «багато до одного», оскільки один вид послуги може фігурувати у багатьох роботах. Опис сутностей із зазначенням їхніх атрибутів і ключових полів наведено у таблиці 2.1.

Таблиця 2.1 – Сутності інфологічної моделі бази даних

| Сутність | Ключовий атрибут | Основні атрибути |
|---|---|---|
| Клієнт | id_client | назва, адреса, телефон, e-mail, відповідальна особа |
| Договір | id_contract | номер договору, дата укладення, дата закінчення, тип обслуговування |
| Комп'ютер | id_computer | інвентарний номер, модель, виробник, дата придбання, місцезнаходження |
| Комплектуюча | id_component | назва, тип, серійний номер, характеристики |
| Програмне забезпечення | id_software | назва, версія, тип ліцензії, дата встановлення, термін ліцензії |
| Заявка | id_request | дата створення, опис проблеми, пріоритет, статус, дата закриття |
| Вид послуги | id_service | назва, опис, нормативний час виконання, вартість |
| Виконана робота | id_work | дата виконання, витрачений час, витрачені матеріали, примітки |

Кількість можливих екземплярів зв'язку між двома сутностями формально описується через кардинальність. Якщо позначити кількість екземплярів сутності $A$, пов'язаних з одним екземпляром сутності $B$, як $K_{AB}$, то для зв'язку «один до багатьох» виконується умова:

$$K_{AB} \geq 1, \qquad K_{BA} = 1 \tag{2.1}$$

де $K_{AB}$ – кількість записів з боку «багато»,

$K_{BA}$ – кількість записів з боку «один».

Ця умова є основоположною для правильного проєктування зовнішніх ключів на наступному етапі [28]. Інфологічна модель у вигляді ER-діаграми наведена на рис. 2.1. Вона відображає всі виділені сутності, їхні ключові атрибути та зв'язки між ними.

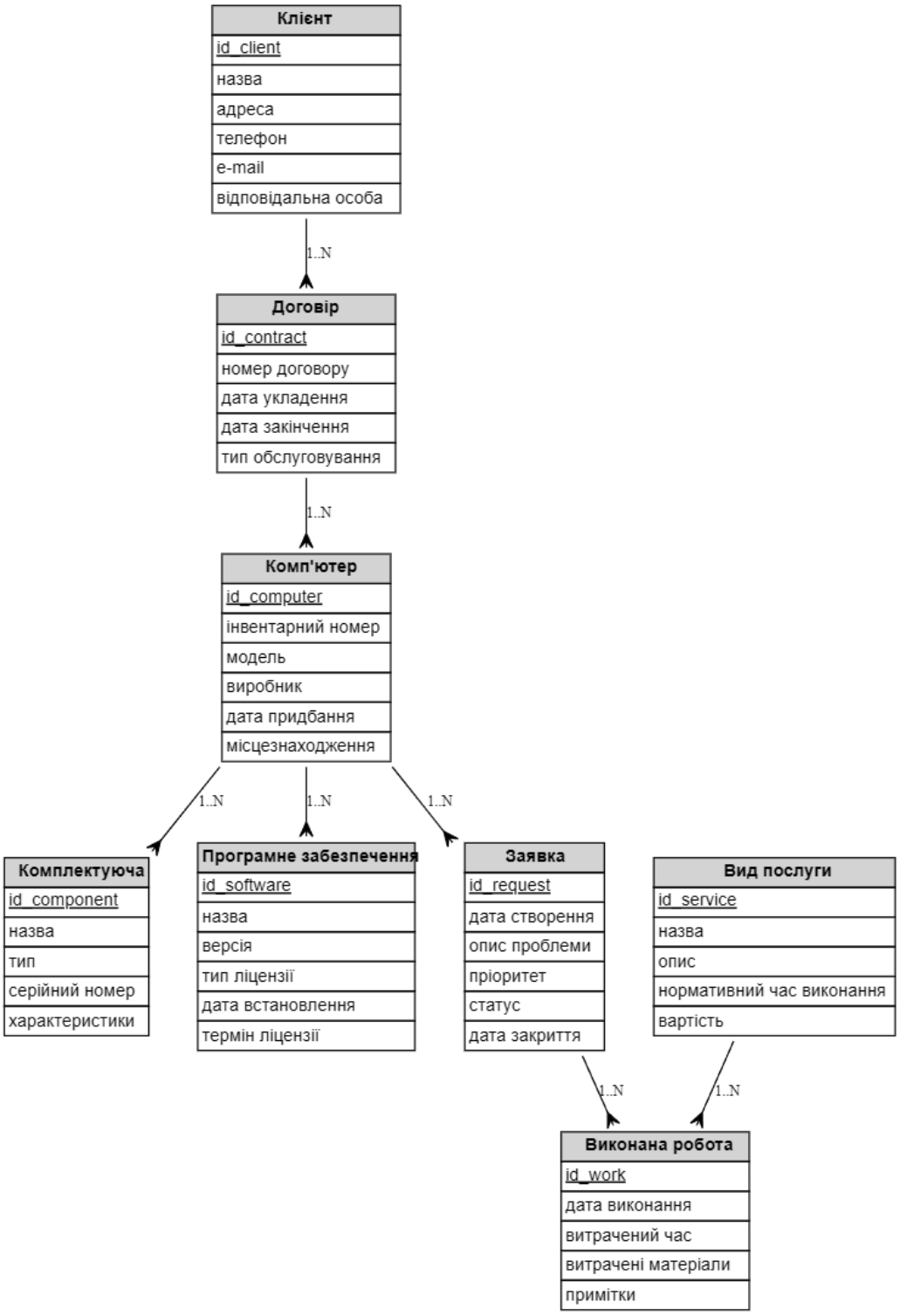


Рисунок 2.1 – ER-діаграма інфологічної моделі бази даних системи

Побудована інфологічна модель є повною, оскільки охоплює всі функціональні вимоги, визначені в підрозділі 1.3: облік клієнтів і договорів, реєстр комп'ютерної техніки до рівня комплектуючих і програмного забезпечення, система заявок із фіксацією виконаних робіт. Вона також є несуперечливою: кожна сутність має унікальний первинний ключ, а зв'язки між сутностями не допускають неоднозначної інтерпретації [29]. Ця модель є вхідним артефактом для логічного проєктування бази даних.

## 2.2 Логічне проєктування бази даних і нормалізація відношень

Логічне проєктування є наступним після концептуального етапом розробки бази даних і полягає у перетворенні інфологічної моделі на схему реляційної бази даних, прив'язану до конкретної моделі даних, але ще незалежну від особливостей конкретної СУБД. Логічне проєктування – це створення схеми бази даних на основі конкретної моделі даних, для реляційної моделі – це набір схем відношень із зазначенням первинних ключів, а також зовнішніх ключів, що представляють зв'язки між відношеннями [26]. Основним інструментом якісного логічного проєктування є нормалізація – процес приведення відношень до нормальних форм з метою усунення надлишковості та аномалій при операціях вставки, оновлення і видалення даних [27].

Правило перетворення ER-моделі у реляційну схему є стандартним: кожній сутності відповідає таблиця, кожному атрибуту – стовпець, ключовий атрибут стає первинним ключем (PK), а зв'язки «один до багатьох» реалізуються через зовнішні ключі (FK) на стороні «багато» [31]. Застосовуючи це правило до інфологічної моделі, побудованої у підрозділі 2.1, отримуємо вісім реляційних таблиць: clients, contracts, computers, components, software, requests, service_types, works.

Центральним поняттям нормалізації є функціональна залежність. Атрибут $B$ функціонально залежить від атрибута $A$ (позначається $A \rightarrow B$), якщо кожному

значенню $A$ відповідає рівно одне значення $B$. Відношення перебуває у третій нормальній формі (3НФ), якщо воно знаходиться у другій нормальній формі (2НФ) і не містить транзитивних функціональних залежностей між неключовими атрибутами [27]. Формально умова відсутності транзитивної залежності записується так:

$$\nexists\, X, Y, Z : X \to Y \wedge Y \to Z \wedge Y \to X \wedge Z \notin X, \tag{2.2}$$

де $X$ – первинний ключ,

$Y$ і $Z$ – неключові атрибути.

Саме виконання цієї умови для всіх таблиць є метою нормалізації в даному проєкті. Розглянемо процес нормалізації на прикладі центральних відношень. Таблиця 2.2 описує повну схему відношень бази даних із зазначенням первинних і зовнішніх ключів та типів даних, які будуть застосовані при фізичній реалізації в MySQL.

Таблиця 2.2 – Схема відношень реляційної бази даних

| Таблиця | Первинний ключ | Зовнішні ключі | Основні атрибути |
|---|---|---|---|
| clients | id_client (INT) | – | name, address, phone, email, contact_person |
| contracts | id_contract (INT) | id_client → clients | contract_number, date_start, date_end, service_type |
| computers | id_computer (INT) | id_contract → contracts | inventory_num, model, manufacturer, purchase_date, location |
| components | id_component (INT) | id_computer → computers | name, type, serial_number, specs |
| software | id_software (INT) | id_computer → computers | name, version, license_type, install_date, expire_date |
| requests | id_request (INT) | id_computer → computers | created_at, description, priority, status, closed_at |
| service_types | id_service (INT) | – | name, description, norm_time, price |
| works | id_work (INT) | id_request → requests; id_service → service_types | work_date, time_spent, materials_used, notes |

Таблиця computers у вихідному (ненормалізованому) варіанті могла б містити поля клієнта безпосередньо у записі комп'ютера, що породжує транзитивну залежність: id_computer → id_contract → id_client → назва клієнта. Після нормалізації дані про клієнта виділені в окрему таблицю clients, а зв'язок проходить через contracts. Таким чином, кожна таблиця у фінальній схемі відповідає 3НФ. Ієрархія зовнішніх ключів точно відображає зв'язки, визначені в інфологічній моделі: clients → contracts → computers – основний ланцюг обліку техніки; computers → {components, software, requests} – деталізація кожного комп'ютера; {requests, service_types} → works – журнал виконаних робіт. Таблиця service_types є довідниковою і не має зовнішніх ключів, що відповідає принципу нормалізації. На рисунку 2.2 зображено схему зв'язків (ERD на рівні логічної моделі) з усіма таблицями, первинними та зовнішніми ключами, що відповідає результатам нормалізації.

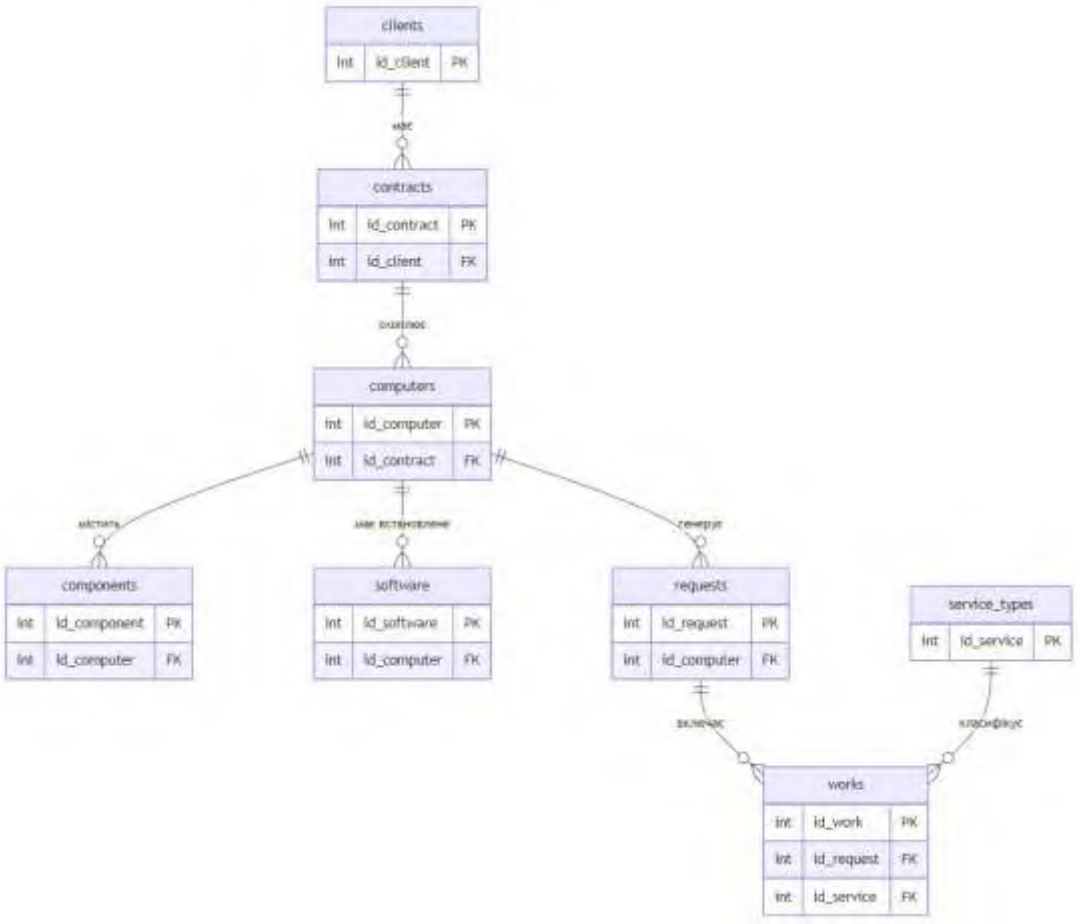


Рисунок 2.2 – Логічна схема реляційної бази даних (після нормалізації)

Отримана логічна схема є несуперечливою, нормалізованою до 3НФ та повністю відповідає функціональним вимогам системи [31]. Відсутність надлишковості гарантує, що зміна, наприклад, адреси клієнта вимагає оновлення лише одного запису в таблиці clients, а не численних записів у кількох таблицях. Така структура мінімізує ризик виникнення аномалій при експлуатації бази даних і є оптимальною основою для фізичної реалізації в MySQL.

## 2.3 Фізична реалізація бази даних у MySQL

Фізичне проєктування є завершальним етапом розробки бази даних і полягає у реалізації логічної схеми засобами конкретної СУБД з урахуванням її синтаксису, типів даних, механізмів забезпечення цілісності та продуктивності. На цьому етапі кожне відношення перетворюється на таблицю MySQL з конкретними типами стовпців, обмеженнями (NOT NULL, UNIQUE, DEFAULT) та зовнішніми ключами.

Для зберігання даних у MySQL обрано рушій InnoDB, який є стандартним для сучасних версій системи і забезпечує підтримку транзакцій, зовнішніх ключів та рядкового блокування – на відміну від застарілого рушія MyISAM [31]. Підтримка зовнішніх ключів є критичною вимогою для підтримання референціальної цілісності бази даних: при спробі видалити клієнта, з яким пов'язані договори, СУБД автоматично відхилить операцію або виконає каскадне оновлення залежно від налаштування параметра ON DELETE [27].

У даній роботі використано кілька важливих проєктних рішень. По-перше, для всіх первинних ключів обрано тип INT UNSIGNED AUTO_INCREMENT – це забезпечує автоматичну генерацію унікальних ідентифікаторів без ризику від'ємних значень. По-друге, статус заявки реалізовано через тип ENUM, що дозволяє на рівні СУБД обмежити множину допустимих значень і унеможливити введення некоректних статусів без додаткової перевірки в коді застосунку. По-третє, для часових витрат використано DECIMAL(5,2), що дозволяє зберігати

значення з точністю до сотих (наприклад, 1.50 год) і уникнути проблем округлення, характерних для типу FLOAT [7]. Таблиця 2.3 містить опис фізичної структури ключових таблиць бази даних із зазначенням типів даних MySQL та обмежень цілісності.

Таблиця 2.3 – Фізична структура основних таблиць бази даних

| Таблиця | Стовпець | Тип MySQL | Обмеження |
|---|---|---|---|
| clients | id_client | INT UNSIGNED | PK, AUTO_INCREMENT |
| clients | name | VARCHAR(200) | NOT NULL |
| clients | phone | VARCHAR(20) | – |
| clients | email | VARCHAR(100) | UNIQUE |
| contracts | id_contract | INT UNSIGNED | PK, AUTO_INCREMENT |
| contracts | id_client | INT UNSIGNED | FK → clients, NOT NULL |
| contracts | date_start | DATE | NOT NULL |
| contracts | date_end | DATE | – |
| computers | id_computer | INT UNSIGNED | PK, AUTO_INCREMENT |
| computers | id_contract | INT UNSIGNED | FK → contracts, NOT NULL |
| computers | inventory_num | VARCHAR(50) | UNIQUE, NOT NULL |
| requests | id_request | INT UNSIGNED | PK, AUTO_INCREMENT |
| requests | id_computer | INT UNSIGNED | FK → computers, NOT NULL |
| requests | status | ENUM('нова','в роботі','закрита') | DEFAULT 'нова' |
| works | id_work | INT UNSIGNED | PK, AUTO_INCREMENT |
| works | id_request | INT UNSIGNED | FK → requests, NOT NULL |
| works | id_service | INT UNSIGNED | FK → service_types, NOT NULL |
| works | time_spent | DECIMAL(5,2) | NOT NULL |

Важливим аспектом фізичної реалізації є створення індексів для оптимізації запитів. Первинні ключі індексуються автоматично. Додатково рекомендується створити індекси на стовпцях зовнішніх ключів та стовпцях, що часто використовуються у виразах WHERE. Зокрема, для таблиці requests доцільно проіндексувати стовпці status та created_at, оскільки типові запити фільтруватимуть заявки саме за цими полями [5].

Ефективність виконання запиту можна оцінити через кількість операцій порівняння. Для таблиці з $n$ рядками без індексу необхідно виконати $O(n)$ порівнянь, тоді як з B-tree-індексом – лише $O(\log_2 n)$ порівнянь:

$$T_{\text{без індексу}} = O(n), \qquad T_{\text{з індексом}} = O(\log_2 n). \tag{2.3}$$

Так, при $n = 10\,000$ записів індекс скорочує кількість порівнянь приблизно з 10000 до 14, що суттєво підвищує швидкодію системи при зростанні обсягу даних [5].

Для підтримання цілісності даних при каскадних операціях налаштування зовнішніх ключів виконується з параметром ON DELETE RESTRICT – це не дозволить видалити батьківський запис, поки існують залежні дочірні записи. Наприклад, клієнт не може бути видалений із бази, якщо з ним пов'язані активні договори. Водночас для стовпців id_client у таблиці contracts встановлено ON UPDATE CASCADE – це дозволяє автоматично оновити зовнішній ключ у дочірніх записах при зміні значення первинного ключа батьківського запису, хоча на практиці первинні ключі типу AUTO_INCREMENT майже ніколи не змінюються [27].

На рисунку 2.3 наведено фрагмент SQL-скрипту створення двох ключових таблиць бази даних, що ілюструє практичну реалізацію описаних вище принципів.

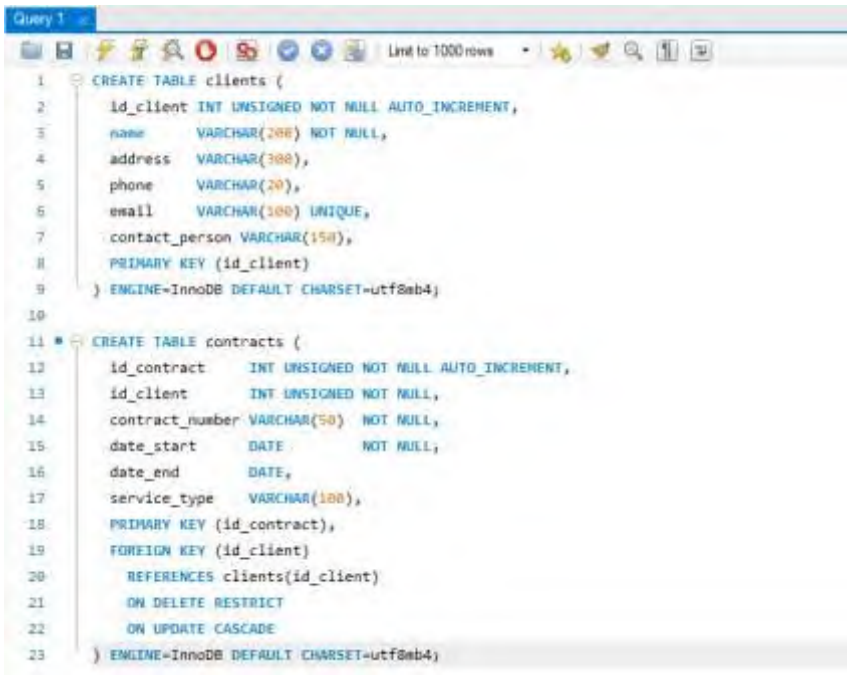


Рисунок 2.3 – Фрагмент SQL-скрипту створення таблиць бази даних

Вибір кодування utf8mb4 є обов'язковим для коректного зберігання україномовних даних, оскільки саме ця кодова сторінка забезпечує повну підтримку Unicode, включно з символами кирилиці та емодзі [27]. Застосування кодування utf8 (у застарілому розумінні MySQL) призводить до проблем з деякими Unicode-символами через обмеження в 3 байти на символ. Фізична реалізація всіх восьми таблиць виконується за аналогічним принципом, що є основою для подальшого розроблення інтерфейсу користувача засобами Python.

## 2.4 Проєктування архітектури програмного інтерфейсу засобами Python

Архітектура програмного інтерфейсу визначає спосіб організації коду застосунку, розподіл відповідальності між його компонентами та механізми взаємодії між шаром відображення даних і шаром доступу до бази даних. Для настільного застосунку на Python, що взаємодіє з MySQL, найбільш доцільним є застосування патерну проєктування MVC (Model-View-Controller – Модель-Вигляд-Контролер), який чітко розмежовує три рівні відповідальності [31].

У рамках MVC-патерну компонент Model відповідає за взаємодію з базою даних: виконання SQL-запитів через бібліотеку mysql-connector-python, отримання результатів та їх передачу у вигляді Python-об'єктів. Компонент View реалізується засобами бібліотеки PyQt5 і відповідає за відображення даних у графічному інтерфейсі: форми, таблиці, діалогові вікна. Компонент Controller містить бізнес-логіку: обробку подій від інтерфейсу, виклик методів моделі та оновлення відображення [21]. Така архітектура полегшує тестування, супровід і подальше розширення системи – зміни в логіці відображення не зачіпають код доступу до даних і навпаки.

Підключення до бази даних реалізується через пул з'єднань, що дозволяє повторно використовувати відкриті сесії MySQL замість створення нового з'єднання при кожному запиті. Бібліотека mysql-connector-python підтримує

параметризовані запити, що є обов'язковим заходом захисту від атак типу SQL-ін'єкція [27]. У параметризованому запиті значення, введені користувачем, передаються окремо від SQL-рядка, і СУБД обробляє їх як дані, а не як частину команди. Таблиця 2.4 відображає розподіл модулів системи за рівнями MVC-архітектури та описує їхні основні функції.

Таблиця 2.4 – Розподіл модулів системи за рівнями MVC-архітектури

| Рівень | Модуль (файл .py) | Основні функції |
|---|---|---|
| Model | db_connection.py | Встановлення та управління підключенням до MySQL |
| Model | clients_model.py | CRUD-операції з таблицею clients |
| Model | computers_model.py | CRUD-операції з таблицями computers, components, software |
| Model | requests_model.py | Створення, оновлення і закриття заявок |
| Model | works_model.py | Реєстрація виконаних робіт, прив'язка до послуг |
| View | main_window.py | Головне вікно з навігаційним меню |
| View | clients_view.py | Форма перегляду і редагування клієнтів |
| View | computers_view.py | Форма реєстру техніки клієнта |
| View | requests_view.py | Форма заявок та їх статусів |
| View | reports_view.py | Форми звітів і фільтрів |
| Controller | clients_controller.py | Обробка подій форми клієнтів |
| Controller | requests_controller.py | Логіка зміни статусів заявок |
| Controller | auth_controller.py | Автентифікація та управління ролями |

Графічний інтерфейс побудований за принципом головного вікна з вкладками (tab widget), де кожна вкладка відповідає функціональному модулю системи: «Клієнти», «Техніка», «Заявки», «Звіти», «Налаштування». Всередині кожної вкладки використовується компонент QTableWidget для відображення табличних даних та QFormLayout для форм введення і редагування. Валідація введених даних реалізується на рівні контролера до передачі даних моделі: перевіряється заповненість обов'язкових полів, формат електронної пошти, коректність дат [22].

Механізм авторизації реалізується через таблицю users бази даних, що зберігає облікові записи з хешованими паролями (алгоритм bcrypt) та ідентифікатором ролі. При запуску застосунку відображається вікно входу, де користувач вводить логін і пароль. Контролер auth_controller порівнює хеш введеного пароля з хешом у базі та, у разі успіху, ініціалізує сесію користувача з

відповідним рівнем доступу [18]. Відповідно до ролі (адміністратор, технічний спеціаліст, менеджер) активуються або деактивуються відповідні пункти меню та кнопки у вигляді. Рисунок 2.4 ілюструє загальну структуру модулів застосунку та потоки взаємодії між ними в рамках MVC-архітектури.

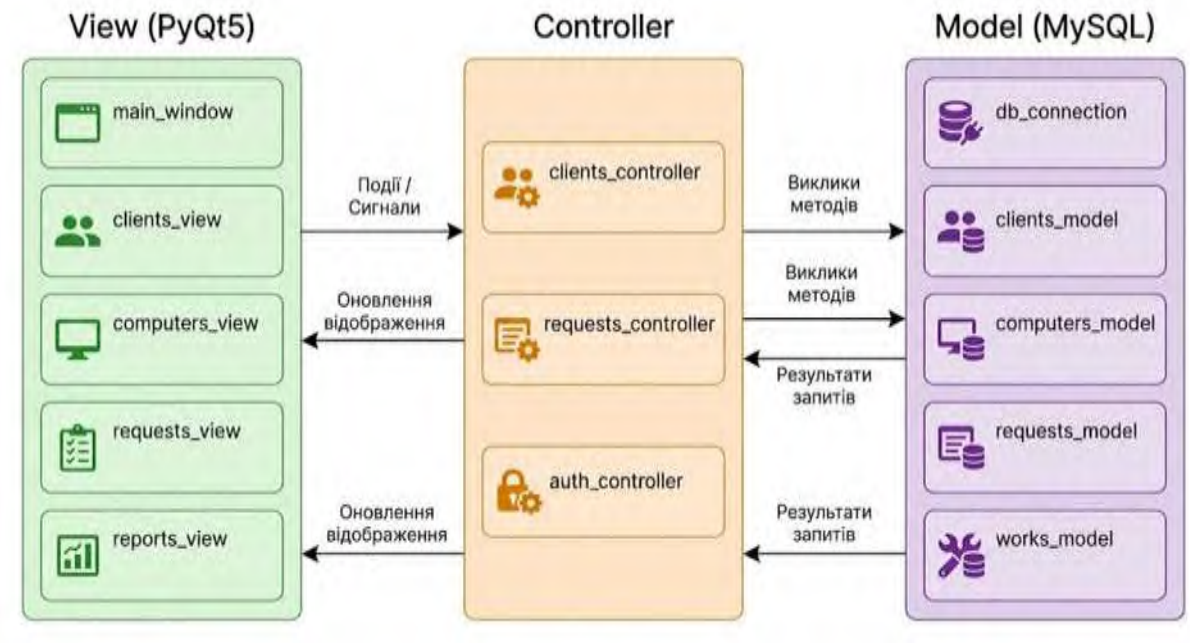


Рисунок 2.4 – Структура модулів застосунку та взаємодія компонентів MVC

Модуль звітності реалізує формування трьох типів звітів: журнал заявок за заданий період, акт виконаних робіт по клієнту та зведена відомість обслуговування техніки. Звіти генеруються засобами бібліотеки reportlab у форматі PDF та можуть бути роздруковані безпосередньо з інтерфейсу або збережені у файл. SQL-запити для звітів використовують JOIN-операції між таблицями clients, contracts, computers, requests та works, що є основною перевагою реляційної моделі даних [5].

Таким чином, виконано повний цикл проєктування бази даних для інформаційної системи обліку комп'ютерної техніки в умовах IT-аутсорсингу: побудовано інфологічну модель предметної області у вигляді ER-діаграми з визначенням восьми ключових сутностей та зв'язків між ними; розроблено логічну схему реляційної бази даних і виконано нормалізацію всіх відношень до

третьої нормальної форми, що забезпечує відсутність надлишковості та аномалій при операціях з даними; здійснено фізичну реалізацію бази даних у MySQL з визначенням типів даних, обмежень цілісності, зовнішніх ключів та індексів; спроєктовано архітектуру програмного інтерфейсу на основі патерну MVC із розподілом модулів на рівні Model, View і Controller та описом механізмів авторизації, валідації та формування звітності. Отримані проєктні рішення є повною і несуперечливою основою для безпосередньої реалізації системи засобами Python і PyQt5, що розглядається у третьому розділі роботи.

# РОЗДІЛ 3
# РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ ЗАСОБАМИ PYTHON І MYSQL ТА ОЦІНЮВАННЯ ЇЇ ЕФЕКТИВНОСТІ

## 3.1 Реалізація інтерфейсу користувача інформаційної системи засобами Python

Графічний інтерфейс користувача (GUI) є ключовою складовою будь-якої настільної інформаційної системи, оскільки саме через нього відбувається вся взаємодія кінцевого користувача з базою даних [22]. Розроблена система обліку комп'ютерної техніки реалізована мовою Python з використанням бібліотеки PyQt5, яка є потужним і зрілим фреймворком для побудови настільних застосунків з багатим набором компонентів – вікон, форм, таблиць, діалогів та меню. Стиль оформлення інтерфейсу встановлено як «Fusion» – сучасна кросплатформна тема, що забезпечує чистий, лаконічний вигляд на операційних системах Windows та Linux.

Точкою входу застосунку є файл main.py, який при запуску ініціює підключення до бази даних MySQL через модуль db.connection. У разі невдалого підключення – наприклад, якщо MySQL Server не запущений або параметри конфігурації у файлі config.py некоректні – користувачеві відображається повідомлення про помилку з конкретними рекомендаціями щодо усунення неполадки, після чого застосунок завершує роботу. Такий підхід забезпечує зрозумілу діагностику на ранньому етапі запуску.

Першим вікном, яке бачить користувач, є вікно авторизації. Воно реалізоване у модулі login_view.py як діалогове вікно фіксованого розміру 360×220 пікселів з полями введення логіну та пароля і кнопками «Увійти» / «Скасувати». Поле пароля працює у захищеному режимі – символи замінюються на маски. Контролер auth_controller отримує введені дані, хешує пароль і порівнює його з хешем, що зберігається у таблиці users бази даних [18]. При

успішній авторизації сесія ініціалізується з роллю користувача (адміністратор, менеджер або технічний спеціаліст), що визначає набір доступних функцій у головному вікні. На рис. 3.1 показано вікно авторизації застосунку.

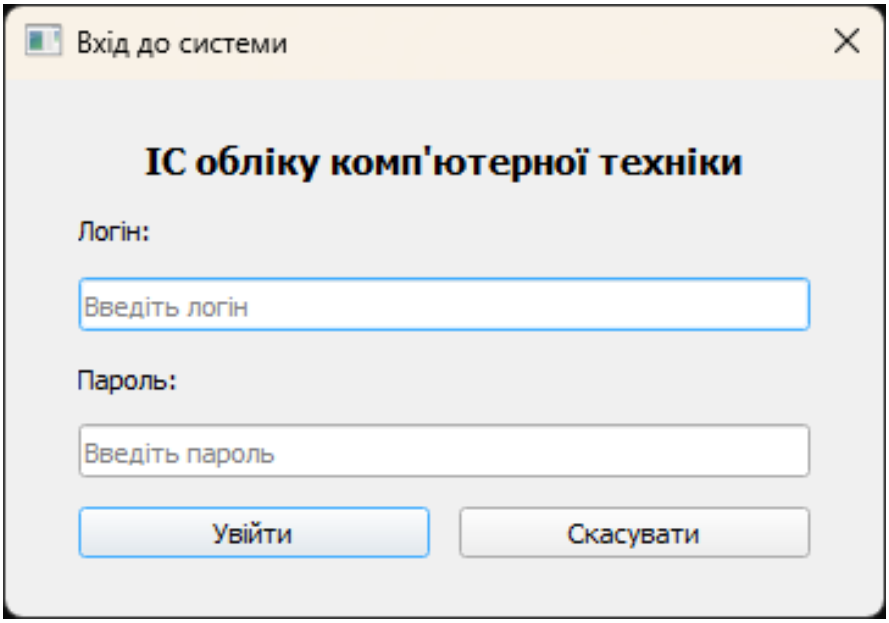


Рисунок 3.1 – Вікно авторизації застосунку

На рис. 3.2 наведено зовнішній вигляд головного вікна застосунку.

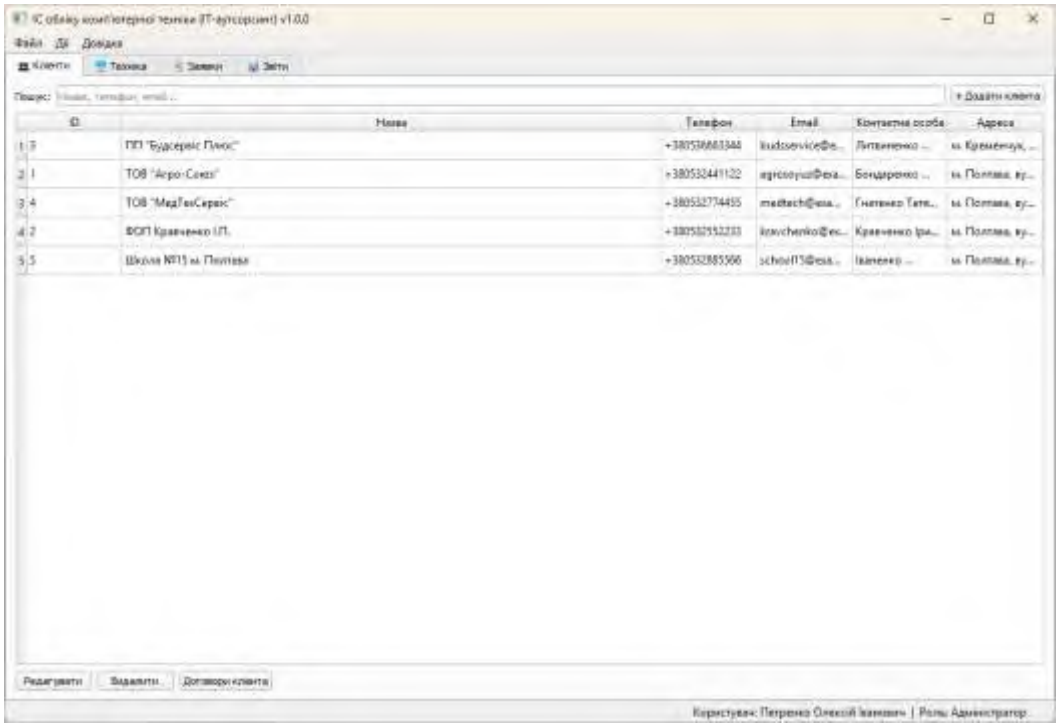

Рисунок 3.2 – Головне вікно застосунку з навігаційними вкладками

Головне вікно застосунку (main_window.py) побудовано на основі компонента QMainWindow з мінімальним розміром 1100×700 пікселів і містить рядок меню, центральний елемент із вкладками та рядок статусу. Рядок меню включає три розділи: «Файл» (вихід із застосунку), «Дії» (оновлення поточної вкладки клавішею F5) та «Довідка» (вікно «Про програму»). У рядку статусу постійно відображається ім'я авторизованого користувача та його роль у системі. Центральна область містить чотири вкладки: «Клієнти», «Техніка», «Заявки» та «Звіти» – кожна з яких відповідає окремому функціональному модулю.

Вкладка «Клієнти» (clients_view.py) є центральним модулем для ведення реєстру підприємств-замовників. У верхній частині вкладки розташована панель пошуку з текстовим полем, що виконує динамічну фільтрацію записів у режимі реального часу в міру введення запиту – пошук охоплює назву, телефон та електронну пошту клієнта. Нижче розміщена таблиця QTableWidget із шістьма стовпцями: ідентифікатор, назва організації, телефон, email, контактна особа та адреса. Стовпець «Назва» розтягується пропорційно ширині вікна. Подвійне клацання на рядку відкриває діалог редагування клієнта. Кнопкова панель містить три дії: «Редагувати», «Видалити» та «Договори клієнта». Остання відкриває окреме діалогове вікно з таблицею договорів обраного клієнта, де відображаються номер договору, дати початку і закінчення, тип послуги та щомісячна оплата. У таблиці 3.1 наведено перелік полів діалогу редагування клієнта.

Таблиця 3.1 – Поля форми редагування клієнта

| Поле форми | Тип компонента | Обов'язкове |
|---|---|---|
| Назва | QLineEdit | Так |
| Адреса | QLineEdit | Ні |
| Телефон | QLineEdit | Ні |
| Email | QLineEdit | Ні |
| Контактна особа | QLineEdit | Ні |
| Нотатки | QTextEdit | Ні |

Вкладка «Заявки» (requests_view.py) реалізує повний цикл обробки сервісних запитів від клієнтів. У верхній частині розміщено фільтр за статусом

заявки (всі / нова / в роботі / закрита). Таблиця заявок містить сім стовпців: ідентифікатор, назву клієнта, ідентифікатор комп'ютера, скорочений опис проблеми, пріоритет, статус та дату створення. Для покращення читабельності рядки таблиці підсвічуються кольором залежно від пріоритету заявки: червоний відтінок для високого, жовтий для середнього і зелений для низького пріоритету. Аналогічне кольорове маркування застосовується і для статусів. Кнопкова панель надає доступ до функцій: «Відкрити» (перегляд і редагування), «Виконані роботи» (перелік робіт по заявці) та «Змінити статус» [6]. На рисунку 3.3 наведено вигляд вкладки «Заявки» з кольоровим маркуванням пріоритетів.

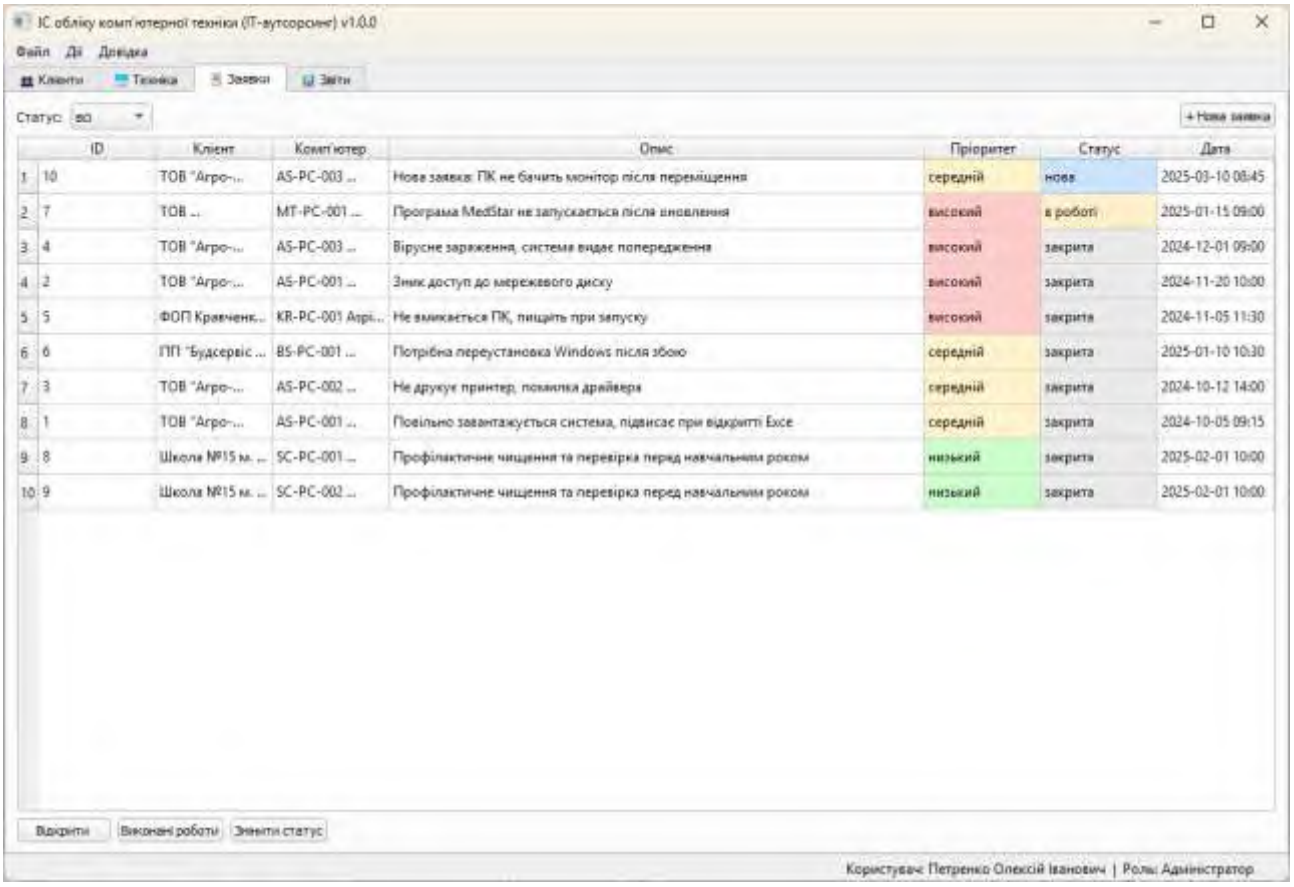


Рисунок 3.3 – Вкладка «Заявки» із кольоровим маркуванням пріоритетів

Вкладка «Звіти» (reports_view.py) забезпечує формування аналітичної звітності. Доступні три типи звітів: журнал заявок за обраний часовий проміжок, акт виконаних робіт у розрізі клієнта та зведена відомість обслуговування техніки. Для вибору дат використовується компонент QDateEdit. Звіти

формуються засобами бібліотеки reportlab у форматі PDF, що дозволяє їх зберегти на диск або роздрукувати безпосередньо з інтерфейсу [13]. SQL-запити для формування звітів реалізують багатотабличні з'єднання (JOIN) між таблицями clients, contracts, computers, requests та works, що є основною перевагою обраної реляційної моделі даних.

Усі форми введення даних реалізують клієнтську валідацію на рівні контролерів: перевіряється заповненість обов'язкових полів до передачі даних у модель [22]. Якщо обов'язкове поле порожнє, користувачеві відображається попереджувальне повідомлення, а збереження блокується. Такий підхід захищає базу даних від потрапляння некоректних або неповних записів ще до виконання SQL-запиту.

## 3.2 Тестування функціональності системи та забезпечення цілісності даних

Тестування інформаційної системи є обов'язковим етапом розробки програмного забезпечення, що дозволяє виявити та усунути помилки до введення системи в експлуатацію, а також підтвердити відповідність реалізованих функцій визначеним вимогам [14]. Для розробленої системи обліку комп'ютерної техніки застосовувалося функціональне тестування – метод перевірки відповідності поведінки системи очікуваним результатам для конкретних вхідних даних і сценаріїв використання [18]. Тестування охопило всі основні функціональні модулі: авторизацію, управління клієнтами, облік техніки, обробку заявок та формування звітів, а також механізми забезпечення цілісності даних на рівні бази даних.

Тестування модуля авторизації передбачало перевірку чотирьох сценаріїв: успішний вхід із коректними обліковими даними, спроба входу з неправильним паролем, введення неіснуючого логіну та залишення поля логіну порожнім. В усіх негативних сценаріях система відображала відповідне повідомлення про

помилку без розкриття деталей (тобто повідомлення «Невірний логін або пароль» є однаковим незалежно від того, чи помилка в логіні, чи в паролі – це унеможливлює перебір облікових записів). Після успішного входу в рядку статусу головного вікна коректно відображалися ім'я та роль авторизованого користувача. Перевірка рольового доступу підтвердила, що технічний спеціаліст не бачить адміністративних функцій управління користувачами, а менеджер не може змінювати технічні характеристики обладнання.

Тестування модуля управління клієнтами включало перевірку операцій створення, читання, оновлення та видалення записів (CRUD). При спробі зберегти нового клієнта з порожньою назвою організації система коректно блокувала збереження та виводила попереджувальне повідомлення – валідація на рівні контролера clients_controller.py спрацювала відповідно до очікувань [22]. Пошук клієнтів у режимі реального часу повернув коректні результати при фільтрації за частковим збігом назви, номером телефону та електронною поштою. Важливим тестом на цілісність даних стала спроба видалити клієнта, з яким пов'язані активні договори: система відмовила у виконанні операції завдяки обмеженню ON DELETE RESTRICT на зовнішньому ключі таблиці contracts, і виведено відповідне повідомлення про причину відмови. Це підтверджує коректне спрацювання декларативних обмежень цілісності на рівні MySQL [27].

Тестування модуля заявок підтвердило коректну роботу фільтрації за статусом – при виборі значення «в роботі» таблиця відображала лише відповідні записи, запити до БД з параметром WHERE status = 'в роботі' виконувалися коректно. Кольорове маркування рядків за пріоритетом і статусом відображалося правильно для всіх можливих значень. Перевірка функції «Виконані роботи» показала, що для кожної заявки коректно підтягуються пов'язані записи з таблиці works через зовнішній ключ id_request. При зміні статусу заявки на «закрита» система фіксує дату закриття та ідентифікатор виконавця – це дозволяє зберегти повну історію обробки кожного звернення.

Окремо перевірялося забезпечення цілісності даних на рівні бази даних. Тест каскадних обмежень підтвердив, що видалення запису з таблиці clients

неможливе за наявності пов'язаних договорів, а видалення договору – за наявності пов'язаних комп'ютерів. Таким чином, ланцюг цілісності clients → contracts → computers → components / software / requests → works повністю захищений від некоректних операцій видалення [27]. Перевірка типів даних показала, що поля з обмеженням NOT NULL не приймають порожніх значень: спроба вставити запис без обов'язкових полів через прямий SQL-запит повертає помилку MySQL ERROR 1048. У таблиці 3.2 наведено результати функціонального тестування основних сценаріїв роботи системи.

Таблиця 3.2 – Результати функціонального тестування системи

| № | Сценарій тестування | Очікуваний результат | Фактичний результат | Статус |
|---|---|---|---|---|
| 1 | Вхід із коректними даними | Відкриття головного вікна | Відкрито головне вікно | Пройдено |
| 2 | Вхід із невірним паролем | Повідомлення про помилку | Повідомлення відображено | Пройдено |
| 3 | Додавання клієнта без назви | Блокування збереження | Збереження заблоковано | Пройдено |
| 4 | Видалення клієнта з договорами | Відмова з поясненням | Відмову отримано | Пройдено |
| 5 | Пошук клієнта за частиною назви | Фільтрація результатів | Фільтрація коректна | Пройдено |
| 6 | Створення заявки без опису | Блокування збереження | Збереження заблоковано | Пройдено |
| 7 | Зміна статусу заявки | Оновлення запису в БД | Запис оновлено | Пройдено |
| 8 | Додавання ПЗ до комп'ютера | Збереження у таблиці software | Запис створено | Пройдено |
| 9 | Формування PDF-звіту | Генерація файлу reportlab | Файл згенеровано | Пройдено |
| 10 | Фільтрація заявок за статусом | Відображення відфільтрованих рядків | Фільтрація коректна | Пройдено |

Тестування формування звітів охопило всі три типи: журнал заявок за вказаний часовий проміжок, акт виконаних робіт по клієнту та зведену відомість обслуговування техніки. У кожному випадку перевірялися коректність вибірки даних (відповідність обраному діапазону дат і клієнту), повнота відображення полів у PDF-документі та можливість зберегти файл на диск. Усі тести

завершилися успішно. На рисунку 3.4 схематично зображено покриття функціональних модулів тестуванням.

| Функціональний модуль | Статус |
|---|---|
| Авторизація | ☑ перевірено |
| Клієнти | ☑ перевірено |
| Техніка | ☑ перевірено |
| Заявки | ☑ перевірено |
| Звіти | ☑ перевірено |

Рисунок 3.4 – Схема покриття функціональних модулів тестуванням

Тестування модуля обліку техніки (computers_view.py) включало перевірку реєстрації нового комп'ютера з прив'язкою до клієнта та договору, додавання компонентів (процесор, ОЗП, накопичувач) до запису обладнання, а також занесення встановленого програмного забезпечення з зазначенням типу ліцензії [15]. Перевірка унікальності інвентарного номера підтвердила, що спроба зареєструвати два комп'ютери з однаковим inventory_num повертає помилку через обмеження UNIQUE у таблиці computers. Це відповідає вимозі до унікальності облікових ідентифікаторів техніки відповідно до рекомендацій щодо обліку комп'ютерного майна [16].

Загалом усі десять тестових сценаріїв, наведених у таблиці 3.2, завершилися з позитивним результатом. Виявлених критичних помилок не зафіксовано. Проведене тестування підтверджує, що розроблена система відповідає функціональним вимогам, сформульованим у підрозділі 1.3, а механізми забезпечення цілісності даних – як програмні (валідація на рівні контролерів), так і декларативні (обмеження зовнішніх ключів у MySQL) – функціонують коректно та надійно захищають базу даних від некоректних операцій [14].

## 3.3 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування є обов'язковим елементом оцінки будь-якого ІТ-проєкту, оскільки дозволяє підтвердити доцільність витрат на розробку та впровадження інформаційної системи шляхом порівняння інвестицій з очікуваними вигодами [2]. Для оцінки ефективності розробленої системи обліку комп'ютерної техніки в умовах ІТ-аутсорсингу застосовано метод розрахунку прямих витрат на розробку та показника повернення інвестицій (ROI), що є найбільш доцільним підходом для проєктів, орієнтованих на зниження трудових витрат і підвищення якості управлінських процесів [2].

Оцінка витрат на розробку системи. Розроблена система реалізована виключно з використанням безкоштовного програмного забезпечення з відкритим кодом: мова програмування Python [11], СУБД MySQL [12], бібліотеки PyQt5, mysql-connector-python та reportlab [13]. Витрати на придбання програмних засобів (ВППЗ) дорівнюють нулю. Основну частину витрат становить оплата праці розробника. Орієнтовна трудомісткість розробки системи наведена у таблиці 3.3.

Таблиця 3.3 – Трудомісткість розробки інформаційної системи

| № | Етап розробки | Трудомісткість, год |
|---|---|---|
| 1 | Аналіз вимог та проєктування БД | 20 |
| 2 | Фізична реалізація бази даних у MySQL | 10 |
| 3 | Розробка модулів Model (Python) | 15 |
| 4 | Розробка графічного інтерфейсу (PyQt5) | 30 |
| 5 | Розробка модуля звітності (reportlab) | 10 |
| 6 | Тестування та налагодження | 15 |
| Разом | | 100 |

Годинна ставка розробника програмного забезпечення рівня Junior/Middle в Україні у 2024–2026 роках становить орієнтовно 200-250 грн/год [6]. Для розрахунку прийнято ставку 220 грн/год. Витрати на оплату праці розробника:

$$\text{ВОП} = T \cdot \text{С}_{\text{год}}, \quad (3.1)$$

де $T$ – трудомісткість розробки (год),

$\text{С}_{\text{год}}$ – годинна ставка розробника (грн/год).

$$\text{ВОП} = 100 \cdot 220 = 22\,000 \text{ грн}$$

Відрахування на соціальні заходи (єдиний соціальний внесок) становлять 22% від фонду оплати праці:

$$\text{ВВСЗ} = \text{ВОП} \cdot 0{,}22 = 22\,000 \cdot 0{,}22 = 4\,840 \text{ грн}$$

Інші прямі витрати (електроенергія, амортизація обладнання розробника) оцінені у 500 грн. Таким чином, загальні прямі витрати на розробку становлять:

$$\text{ВП} = \text{ВОП} + \text{ВВСЗ} + \text{ВІ} = 22\,000 + 4\,840 + 500 = 27\,340 \text{ грн}$$

Непрямі витрати (ВН) для даного проєкту є мінімальними, оскільки розробка виконувалась індивідуально без виробничих простоїв. Щорічні витрати на утримання системи (ВУТР) включають лише час на оновлення та технічну підтримку – орієнтовно 5-10 год/рік, що становить 1100-2200 грн/рік.

Оцінка вигод від впровадження системи. До прямих економічних вигод належить економія робочого часу фахівця аутсорсингової компанії, що раніше витрачався на ручне ведення обліку. За оцінками, без автоматизованої системи технічний спеціаліст витрачає в середньому 2 години на тиждень лише на облікові операції (пошук інформації про техніку клієнтів, формування актів, складання звітів). При впровадженні системи цей час скорочується до 20-30 хвилин. Економія часу складає приблизно 1,5 год/тиж., або 78 год/рік. При годинній ставці спеціаліста 200 грн/год річна економія витрат на оплату праці становить $\text{Е}_{\text{рік}} = 78 \cdot 200 = 15\,600 \text{ грн/рік}$.

Якісні ефекти від впровадження включають: підвищення точності обліку завдяки усуненню людського фактору при веденні записів; скорочення часу реакції на заявки клієнтів завдяки миттєвому доступу до актуальної інформації про стан техніки; зниження ризику втрати даних через централізоване зберігання у MySQL з підтримкою резервного копіювання [18]; можливість масштабування

клієнтської бази без пропорційного збільшення адміністративного навантаження.

Розрахунок показника ROI. Показник бухгалтерської рентабельності інвестицій розраховується як відношення середньорічного чистого прибутку (економії) до обсягу інвестицій:

$$ROI = \frac{\text{Е}_{\text{рік}}}{\text{ВП}} \cdot 100\% = \frac{15\,600}{27\,340} \cdot 100\% \approx 57{,}1\% \qquad (3.2)$$

Термін окупності проєкту складає приблизно 21 місяць, що є прийнятним показником для інструментальних IT-проєктів подібного масштабу. На рис. 3.5 наведено структуру витрат на розробку системи у відсотковому виразі.

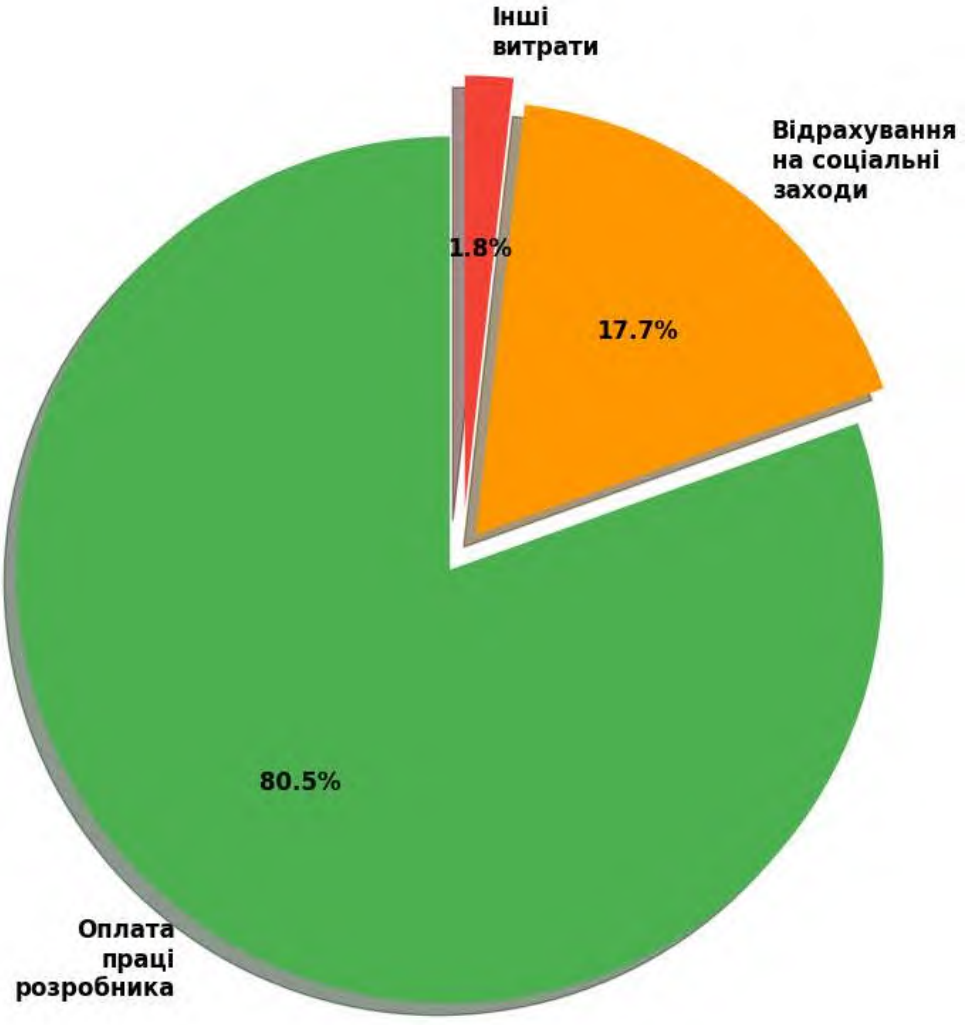


Рисунок 3.5 – Структура витрат на розробку інформаційної системи

Таблиця 3.4 – Зведені техніко-економічні показники проєкту

| Показник | Значення |
|---|---|
| Загальні витрати на розробку (ВП), грн | 27340 |
| Щорічні витрати на утримання (ВУТР), грн/рік | 1650 |
| Річна економія витрат праці (Е_рік), грн/рік | 15600 |
| Показник рентабельності інвестицій (ROI), % | 57,1% |
| Орієнтовний термін окупності, міс. | 21 |

Отримане значення ROI на рівні 57,1% (табл. 3.4) є вищим за орієнтовну ставку банківського депозиту, що підтверджує економічну доцільність розробки та впровадження системи [2]. Водночас слід враховувати, що наведений розрахунок відображає лише прямий ефект від економії трудових витрат. Якісні та стратегічні ефекти – підвищення задоволеності клієнтів, зниження ризику втрати облікових даних, можливість обслуговування більшої кількості клієнтів без збільшення штату – формують додаткову цінність, яка важко піддається прямому кількісному вимірюванню, але суттєво впливає на конкурентоспроможність аутсорсингової компанії на ринку [6].

Таким чином, було виконано практичну реалізацію та оцінювання інформаційної системи обліку комп'ютерної техніки: розроблено повнофункціональний графічний інтерфейс користувача засобами Python і PyQt5 з модулями авторизації, управління клієнтами, обліку техніки, обробки заявок та формування PDF-звітів; проведено функціональне тестування десяти ключових сценаріїв роботи системи, яке підтвердило коректність реалізації всіх функціональних вимог та надійність механізмів забезпечення цілісності даних як на програмному рівні (валідація у контролерах), так і на рівні СУБД (обмеження зовнішніх ключів); здійснено техніко-економічне обґрунтування, яке показало, що загальні витрати на розробку системи становлять 27340 грн, річна економія трудових витрат – 15600 грн, а показник рентабельності інвестицій ROI досягає 57,1% при терміні окупності близько 21 місяця, що підтверджує економічну доцільність створення та впровадження розробленої системи.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнута її початкова мета та вирішені всі поставлені завдання. Проаналізовано основну наукову, методичну та нормативну літературу з теми проєктування баз даних та розробки інформаційних систем для автоматизації обліку комп'ютерної техніки в умовах IT-аутсорсингу.

Проведено аналіз особливостей IT-аутсорсингу в Україні та потреб автоматизації обліку комп'ютерної техніки. Встановлено, що IT-послуги посідають стратегічне місце в українській економіці, а основними споживачами аутсорсингових послуг є підприємства малого та середнього бізнесу, які не можуть утримувати власний штат IT-спеціалістів. Здійснено огляд існуючих програмних рішень – GLPI, OCS Inventory, Snipe-IT, InForm PC – та виявлено їх суттєві обмеження: відсутність орієнтації на модель багатоклієнтського аутсорсингу, недостатня україномовна локалізація та складність розгортання. Це обґрунтувало доцільність власної розробки. Визначено функціональні вимоги до системи (облік клієнтів і договорів, реєстр техніки та компонентів, управління заявками, контроль ліцензій, звітність) та нефункціональні вимоги (продуктивність, безпека, масштабованість, зручність інтерфейсу). Обґрунтовано вибір Python, PyQt5 та MySQL як оптимального технологічного стеку для реалізації системи.

Виконано повний цикл проєктування бази даних. Побудовано інфологічну модель предметної області у вигляді ER-діаграми з визначенням восьми ключових сутностей: clients, contracts, computers, components, software, requests, works та users. Розроблено логічну схему реляційної бази даних і проведено нормалізацію всіх відношень до третьої нормальної форми, що забезпечує відсутність надлишковості та аномалій при операціях з даними. Здійснено фізичну реалізацію бази даних у MySQL 8.4 з визначенням типів даних, обмежень цілісності, зовнішніх ключів з параметрами ON DELETE RESTRICT та індексів для оптимізації запитів. Спроєктовано архітектуру програмного

інтерфейсу на основі патерну MVC із чітким розподілом модулів на рівні Model, View і Controller, описом механізмів авторизації з хешуванням паролів (bcrypt), валідації введених даних та формування звітності у форматі PDF.

Реалізовано повнофункціональний графічний інтерфейс користувача засобами Python і PyQt5, який включає модуль авторизації з рольовим доступом (адміністратор, менеджер, технічний спеціаліст), вкладки управління клієнтами та договорами, обліку комп'ютерної техніки з компонентами та програмним забезпеченням, обробки сервісних заявок із кольоровим маркуванням пріоритетів та модуль формування PDF-звітів. Проведено функціональне тестування десяти ключових сценаріїв роботи системи, яке підтвердило коректність реалізації всіх функціональних вимог та надійність механізмів цілісності даних на рівні СУБД і програмних контролерів.

Техніко-економічне обґрунтування показало, що загальні витрати на розробку системи становлять 27340 грн, річна економія трудових витрат аутсорсингової компанії – 15600 грн, а показник рентабельності інвестицій ROI досягає 57,1% при терміні окупності близько 21 місяця. Це підтверджує економічну доцільність створення та практичного впровадження розробленої системи.

Таким чином, поставлені завдання вирішені у повному обсязі. Напрямками подальших досліджень є розширення функціональності системи шляхом додавання вебінтерфейсу для клієнтів, що дозволить їм самостійно подавати заявки в онлайн-режимі; інтеграція з системами моніторингу мережевої інфраструктури для автоматичного виявлення несправностей; впровадження модуля автоматичних сповіщень про завершення гарантійних термінів та заплановані технічні обслуговування; а також розгортання системи у хмарному середовищі для забезпечення доступу до неї з будь-якого пристрою.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке IT-аутсорсинг? URL: https://cbto.com.ua/library/it-outsorcing (дата звернення: 08.05.2026).

2. Двуліт З., Налутка П. Розвиток IT-аутсорсингу та його вплив на економіку України. *Економіка та суспільство*. 2021. № 34. DOI: https://doi.org/10.32782/2524-0072/2021-34-11 (дата звернення: 08.05.2026).

3. IT-аутсорсинг, обслуговування комп'ютерів та серверів. URL: https://duxit.ua/ (дата звернення: 08.05.2026).

4. Програма для обліку комп’ютерів в офісах та IT компаніях. URL https://eqman.co/uk/programa_dlya_obliku_kompyuteriv_v_ofis/ (дата звернення: 08.05.2026).

5. Малий бізнес України у 2024. URL: https://blog.youcontrol.market/malii-biznies-ukrayini-u-2024/ (дата звернення: 08.05.2026).

6. IT-сектор в Україні: стан ринку наприкінці 2024 року. URL: https://itedu.center/ua/blog/articles/it-market-in-ukraine-end-of-2024/ (дата звернення: 08.05.2026).

7. Smart IT Service Management, Helpdesk & Asset Tracking. GLPI. URL: https://glpi-project.org/ (дата звернення: 08.05.2026).

8. Облік обладнання з OCS Inventory NG і GLPI. URL: http://chaspik.dn.ua/stati/oblik/uk/agent-oblik-obladnanna-z-ocs-inventory-ng-i-glpi/ (дата звернення: 08.05.2026).

9. Eqman тарифи. Оберіть план, який працюватиме на Вас. URL: https://eqman.co/uk/pricing/ (дата звернення: 08.05.2026).

10. Рекомендації щодо забезпечення правомірності використання комп'ютерних програм у діяльності суб'єктів господарювання. URL: https://me.gov.ua/Documents/Detail?lang=uk-UA&id=7e6aafcd-2927-42fb-8e0b-0773863e98ba&title=RekomendatsiiSchodoZabezpechenniaPravomirnostiVikoristanniaKompiuternikhProgramUDiialnostiSubktivGospodariuvannia (дата звернення: 08.05.2026).

11. Де використовується Python: огляд найпопулярніших сфер. URL: https://foxminded.ua/de-vykorystovuietsia-python/ (дата звернення: 08.05.2026).

12. Система управління базами даних MySQL. URL: https://lemon.school/blog/systema-upravlinnya-bazamy-danyh-mysql (дата звернення: 08.05.2026).

13. Повний практичний посібник з Python і MySQL. URL: https://robotdreams.cc/uk/blog/484-python-ta-mysql-povniy-praktichniy-posibnik-ch-1 (дата звернення: 08.05.2026).

14. Функціональні та нефункціональні вимоги: структура, SRS, приклади. URL: https://wezom.com.ua/ua/blog/funktsionalni-ta-nefunktsionalni-vimogi-do-programnogo-zabezpechennya (дата звернення: 08.05.2026).

15. Про платіж за використання комп'ютерної програми. URL: https://zakon.rada.gov.ua/rada/show/v3064872-15#Text

16. Форма. Картка обліку персонального комп’ютера. URL https://www.zoda.gov.ua/files/WP_Article_File/original/000086/86926.pdf (дата звернення: 08.05.2026).

17. Нефункціональні вимоги: продуктивність, доступність, безпека. URL: https://careers.epam.ua/blog/non-functional-requirements-how-to-ensure-performance-availability-and-security (дата звернення: 08.05.2026).

18. ДСТУ ISO/IEC 27001:2015 Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Київ: УкрНДНЦ, 2015. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=66910 (дата звернення: 08.05.2026).

19. Загальна характеристика та класифікація інформаційних систем обліку. URL: https://buklib.net/books/22551/ (дата звернення: 08.05.2026).

20. Що таке аналіз вимог? Процес і методи. URL: https://visuresolutions.com/uk/alm-guide /аналіз-вимог-а/ (дата звернення: 08.05.2026).

21. Python – мова програмування 2024 року за версією TIOBE. URL: https://dev.ua/news/python-stav-naipopuliarnishoiu-movoiu-prohramuvannia-za-indeksom-tiobe-yaki-movy-lyshylysia-pozadu-i-iak-zminylasia-dynamika-za-30-rokiv-1736365443 (дата звернення: 08.05.2026).

22. Python: Графічний інтерфейс користувача (tkinter). URL: https://dystosvita.org.ua/mod/page/view.php?id=755 (дата звернення: 08.05.2026).

23. Босько В.В., Константинова Л.В., Поліщук Л.І., КонопліцькаСлободенюк О.К. Бази даних: Навчальний посібник. Кропивницький: ЦНТУ, 2024. 226 с. URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/5945ba1d-0c97-4b92-a49b-6dbe318853fb/content (дата звернення: 08.05.2026).

24. Типи баз даних: особливості, відмінності та приклади. URL: https://dou.ua/lenta/articles/types-of-databases/ (дата звернення: 08.05.2026).

25. База даних MySQL. URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 08.05.2026).

26. Проєктування бази даних. URL: https://uk.wikipedia.org/wiki/Проєктування_бази_даних (дата звернення: 08.05.2026).

27. Основні етапи проєктування баз даних. URL: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture01 (дата звернення: 08.05.2026).

28. Стрельбіцький М., Мельник А. Проєктування структури бази даних інформаційно-комунікаційної системи обліку персоналу. *Збірник наукових праць Національної академії Державної прикордонної служби України. Серія: військові та технічні науки*. 2025. Т. 99, № 2. С. 134–141. URL: https://doi.org/10.32453/3.v99i2.1865 (дата звернення: 08.05.2026).

29. Модель «сутність – зв'язок». URL: https://uk.wikipedia.org/wiki/Модель_«сутність_–_зв‘язок» (дата звернення: 08.05.2026).

30. Нормальні форми реляційних баз даних. URL: https://uk.wikipedia.org/wiki/Нормальна_форма (дата звернення: 08.05.2026).

31. Робота з базами даних в Python: SQL та NoSQL рішення. URL: https://mate.academy/blog/python/sql-nosql-databases-python (дата звернення: 08.05.2026).