Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

Інформаційна система для оптимізації діяльності підприємства з продажу та обслуговування комп'ютерної техніки

Виконав:
здобувач IV курсу
групи КН-41
спеціальності 122 «Комп'ютерні науки»
Довгаль Владислав Володимирович
Науковий керівник:
доц. Крайчук С.О.

Рівне – 2024

РЕФЕРАТ

Пояснювальна записка: 95 с., 10 табл., 47 рис., 2 дод., 23 джерела.

АВТОМАТИЗАЦІЯ, ПРОДАЖ, ОБСЛУГОВУВАННЯ, КОМП'ЮТЕРНА ТЕХНІКА, ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ.

Об’єкт дослідження: процес продажу та обслуговування комп'ютерної техніки на підприємстві «ТехноСервіс».

Мета роботи полягає у розробці інформаційної системи для оптимізації діяльності підприємства з продажу та обслуговування комп'ютерної техніки.

Методи дослідження – аналіз існуючих моделей продаж та програмних рішень, проектування програмного забезпечення, моделювання бізнес-процесів, тестування функціональних та нефункціональних вимог системи.

В роботі розглянуто актуальні питання автоматизації продажу та обслуговування комп'ютерної техніки, запропоновані рішення для оптимізації процесів взаємодії з клієнтами та управління запасами. Розроблено програмне забезпечення, яке включає модулі для управління продажами, обліком ремонтів, веденням довідників та генерацією звітів. Проведено тестування системи, що підтвердило її функціональність та надійність.

Розроблена система може використовуватись у підприємствах, що займаються продажем та обслуговуванням комп'ютерної техніки, а також у навчальному процесі для підготовки фахівців у сфері автоматизації бізнес-процесів та інформаційних технологій.

ЗМІСТ

ВСТУП

У сучасному світі, який характеризується швидкими темпами розвитку технологій та зростанням конкуренції, підприємства, що займаються продажем та обслуговуванням комп'ютерної техніки, стикаються з численними викликами. Серед них виділяються необхідність забезпечення високої якості обслуговування клієнтів, оптимізація внутрішніх бізнес-процесів, а також ефективне управління запасами і ресурсами.

Актуальність впровадження інформаційних систем у діяльність підприємств з продажу та обслуговування комп'ютерної техніки обумовлюється кількома ключовими факторами. По-перше, такі системи дозволяють автоматизувати рутинні процеси, зменшуючи тим самим навантаження на персонал та мінімізуючи людський фактор у виконанні завдань. Це забезпечує підвищення точності та швидкості обробки даних, що, в свою чергу, сприяє покращенню якості обслуговування клієнтів. По-друге, інформаційні системи надають можливість ефективного управління запасами, що дозволяє уникнути надлишкових витрат на зберігання та закупівлю товарів, а також забезпечити своєчасне поповнення складів.

Окрім того, впровадження інформаційних систем дозволяє підприємствам оптимізувати процеси продажу та обслуговування техніки, що включає управління замовленнями, обробку запитів на обслуговування, контроль за виконанням робіт та моніторинг задоволеності клієнтів. Це, в свою чергу, сприяє підвищенню лояльності клієнтів та збільшенню обсягів продажу. Важливо зазначити, що завдяки використанню інформаційних систем, підприємства отримують можливість аналізувати великі обсяги даних про свою діяльність, що дозволяє виявляти слабкі місця та приймати обґрунтовані рішення щодо покращення бізнес-процесів.

Мета роботи полягає у розробці інформаційної системи для оптимізації діяльності підприємства з продажу та обслуговування комп'ютерної техніки.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз принципів організації та функціонування підприємства, а також дослідити існуючі моделі продаж і взаємодії з клієнтом для виявлення основних проблем і викликів;

– визначити та обґрунтувати вибір технологій для реалізації інформаційної системи, враховуючи сучасні тенденції та специфіку галузі;

– розробити проект інформаційної системи, що включає моделювання основних бізнес-процесів, побудову діаграм прецедентів, проектування структури бази даних та визначення архітектури системи;

– розробити програмне забезпечення, провести функціональне та модульне тестування, а також підготувати інструкцію для користувачів програми для забезпечення ефективного використання системи.

Ці задачі дозволять забезпечити комплексний підхід до розробки інформаційної системи, яка відповідатиме потребам підприємства та сприятиме його успішній діяльності на ринку.

Об’єкт дослідження: процес продажу та обслуговування комп'ютерної техніки на підприємстві «ТехноСервіс».

Предмет дослідження – це інформаційні технології та системи, які застосовуються для оптимізації діяльності підприємства, зокрема методи, інструменти та засоби для автоматизації бізнес-процесів.

Практичне значення одержаних результатів полягає в оптимізації діяльності підприємств з продажу та обслуговування комп'ютерної техніки за рахунок автоматизації бізнес-процесів. Це сприятиме підвищенню ефективності обслуговування клієнтів, зменшенню кількості помилок, покращенню управління запасами та ресурсами, а також збільшенню аналітичних можливостей підприємств, що в свою чергу забезпечить їх конкурентоспроможність на ринку.

1 АНАЛІЗ ОРГАНІЗАЦІЇ, ФУНКЦІОНУВАННЯ ПІДПРИЄМСТВА ТА ДОСЛІДЖЕННЯ МОДЕЛІ ПРОДАЖ ТА ВЗАЄМОДІЇ З КЛІЄНТОМ

1.1 Огляд принципів організації підприємства

Підприємство «ТехноСервіс» є одним з провідних гравців на ринку продажу та обслуговування комп'ютерної техніки в Україні. Засноване у 2010 році, воно спеціалізується на продажу різноманітної комп'ютерної техніки, включаючи персональні комп'ютери, ноутбуки, сервери, периферійні пристрої та аксесуари, а також на наданні послуг з їх обслуговування та ремонту. За роки своєї діяльності «ТехноСервіс» зарекомендувало себе як надійний партнер для багатьох приватних та корпоративних клієнтів, забезпечуючи високий рівень обслуговування та широкий асортимент продукції.

Організаційна структура підприємства «ТехноСервіс» представлена на рис. 1.1.

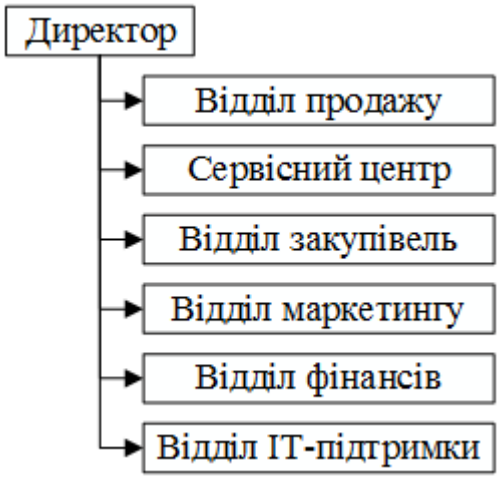


Рисунок 1.1 – Організаційна структура підприємства «ТехноСервіс»

Організаційна структура «ТехноСервіс» складається із:

– директора підприємства «ТехноСервіс» відповідає за стратегічне планування та загальне керівництво компанією, а також за прийняття ключових управлінських рішень, забезпечення ефективної роботи всіх відділів і підрозділів, та підтримку партнерських відносин з ключовими клієнтами та постачальниками;

– відділ продажу – відповідає за продаж комп'ютерної техніки та аксесуарів. До складу цього відділу входять менеджери з продажу, консультанти та спеціалісти з логістики;

– сервісний центр – займається обслуговуванням та ремонтом комп'ютерної техніки. Включає технічних спеціалістів, інженерів та майстрів з ремонту;

– відділ закупівель – забезпечує своєчасну закупівлю необхідних товарів та запасних частин. Відповідає за взаємодію з постачальниками;

– відділ маркетингу – відповідає за просування продукції та послуг, організацію рекламних кампаній, а також за аналіз ринку та конкурентів;

– відділ фінансів – займається фінансовим плануванням, обліком та звітністю. Включає бухгалтерів та фінансових аналітиків;

– відділ IT-підтримки – забезпечує безперебійну роботу інформаційних систем підприємства, підтримує внутрішню IT-інфраструктуру.

Рис. 1.2 відображає основні принципи організації роботи «ТехноСервіс».

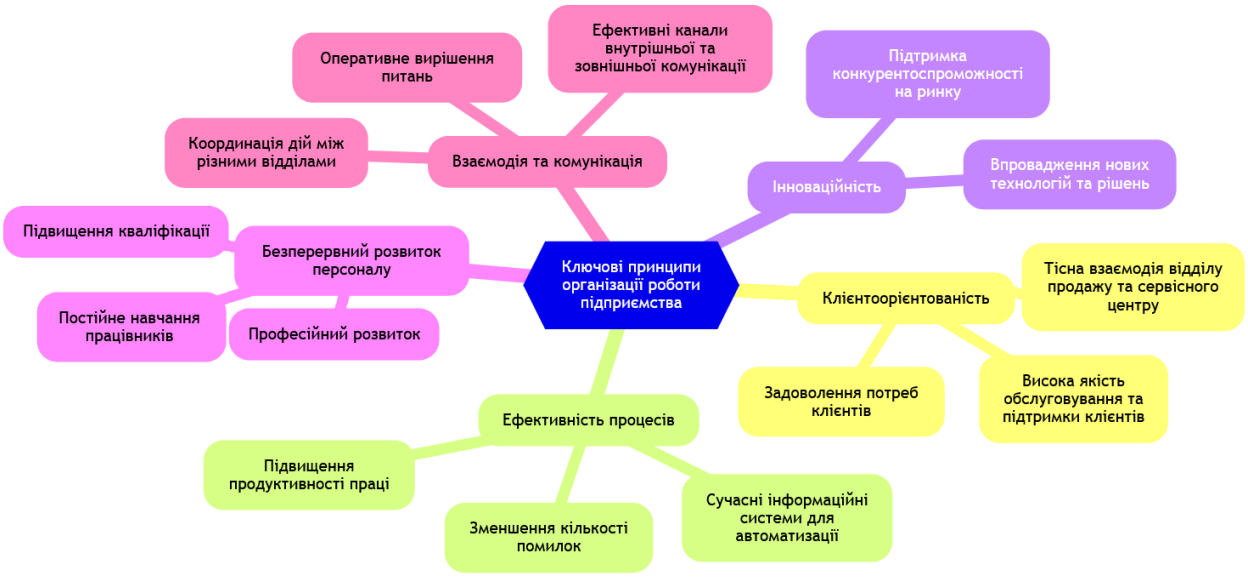


Рисунок 1.2 – Основні принципи організації роботи «ТехноСервіс»

Організація роботи в «ТехноСервіс» базується на таких ключових принципах:

– клієнтоорієнтованість – усі процеси на підприємстві орієнтовані на задоволення потреб клієнтів. Відділ продажу та сервісний центр працюють у

тісній взаємодії, щоб забезпечити високу якість обслуговування та підтримки клієнтів;

– ефективність процесів – впроваджуються сучасні інформаційні системи для автоматизації бізнес-процесів, що дозволяє зменшити кількість помилок та підвищити продуктивність праці;

– інноваційність – підприємство постійно впроваджує нові технології та рішення, що дозволяє підтримувати конкурентоспроможність на ринку.

– безперервний розвиток персоналу – здійснюється постійне навчання та підвищення кваліфікації працівників, що сприяє їх професійному розвитку та підвищенню якості обслуговування;

– взаємодія та комунікація – на підприємстві налагоджені ефективні канали внутрішньої та зовнішньої комунікації, що забезпечує оперативне вирішення питань та координацію дій між різними відділами.

Організаційна структура та принципи роботи підприємства «ТехноСервіс» дозволяють ефективно здійснювати продаж та обслуговування комп'ютерної техніки, забезпечуючи високу якість обслуговування клієнтів та оптимізацію внутрішніх бізнес-процесів. Використання сучасних інформаційних технологій та постійне вдосконалення процесів сприяє підвищенню конкурентоспроможності підприємства на ринку.

1.2 Аналіз функціонування підприємства в сучасних умовах

Підприємство «ТехноСервіс» функціонує в умовах жорсткої конкуренції на ринку продажу та обслуговування комп'ютерної техніки. Конкуренція змушує постійно вдосконалювати асортимент продукції, підвищувати якість обслуговування та впроваджувати сучасні технології для задоволення потреб клієнтів.

«ТехноСервіс» спеціалізується на продажу широкого асортименту комп'ютерної техніки, включаючи персональні комп'ютери, ноутбуки, сервери, периферійні пристрої та аксесуари. Окрім цього, підприємство надає послуги з технічного обслуговування та ремонту комп'ютерної техніки.

Клієнтами компанії є як приватні особи, так і корпоративні клієнти, яким пропонуються комплексні рішення з обслуговування IT-інфраструктури.

В результаті аналізу функціонування підприємства «ТехноСервіс» було виявлено ряд проблеми, з якими воно зіштовхується (рис. 1.3).

Рисунок 1.3 – Основні проблеми підприємства

Проблеми, з якими зіштовхується підприємство «ТехноСервіс» складаються із:

– управління запасами. Однією з основних проблем є ефективне управління запасами. Нестача товарів може призвести до втрати клієнтів, тоді як надлишкові запаси збільшують витрати на їх зберігання. Для оптимізації цього процесу необхідно впровадити сучасні системи управління запасами, що дозволять точно прогнозувати потреби та підтримувати оптимальний рівень запасів;

– автоматизація бізнес-процесів. Відсутність належного рівня автоматизації може призводити до зниження ефективності роботи підприємства та збільшення кількості помилок. Автоматизація процесів прийому замовлень, обробки запитів на обслуговування, управління фінансами та інших операцій сприятиме підвищенню продуктивності та зниженню операційних витрат;

– забезпечення якості обслуговування. Підтримка високого рівня обслуговування клієнтів є ключовим фактором для збереження конкурентоспроможності. Підприємство зіштовхується з проблемами, пов'язаними з недостатньою швидкістю реагування на запити клієнтів та відсутністю систематичного моніторингу якості послуг. Впровадження CRM-системи допоможе в управлінні взаємовідносинами з клієнтами та покращить якість обслуговування;

– інтеграція з існуючими системами. Інтеграція нових інформаційних систем з існуючими може бути складним завданням, що потребує значних ресурсів та часу. Необхідно забезпечити безшовну інтеграцію, щоб уникнути дублювання даних та забезпечити безперебійну роботу всіх бізнес-процесів.

Таким чином, функціонування підприємства «ТехноСервіс» у сучасних умовах потребує постійного вдосконалення та адаптації до нових викликів. Вирішення зазначених проблем шляхом впровадження сучасних інформаційних технологій сприятиме підвищенню ефективності роботи та конкурентоспроможності підприємства.

1.3 Огляд існуючих моделей продаж

Для підприємства «ТехноСервіс», яке спеціалізується на продажу та обслуговуванні комп'ютерної техніки, розробка додатку може значно оптимізувати бізнес-процеси та підвищити ефективність роботи. Нижче розглянуто кілька існуючих моделей продажу, які можуть бути інтегровані з додатком для вирішення комплексних проблем підприємства.

1. Модель прямих продаж (Direct Sales Model)

Модель прямих продаж передбачає безпосередню взаємодію підприємства з клієнтами без посередників. Додаток служить основним інструментом для оформлення замовлень, управління клієнтськими заявками та підтримки зв'язку з клієнтами.

На рис. 1.4 представлено діаграму послідовності моделі прямих продаж.

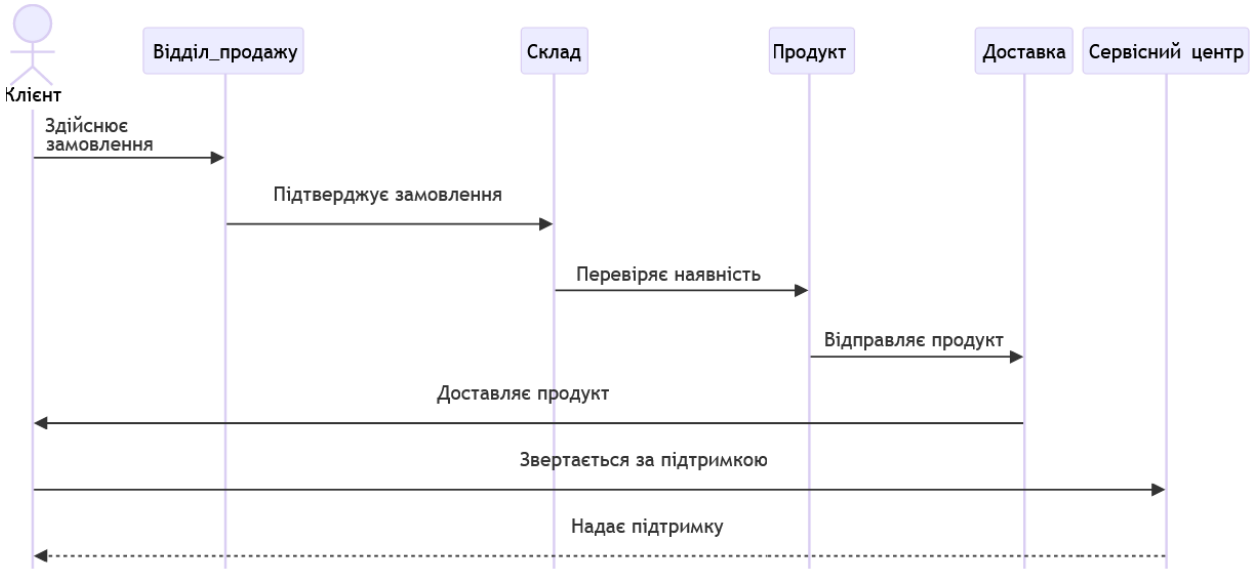


Рисунок 1.4 – Модель прямих продаж

У цій моделі клієнт здійснює замовлення через відділ продажу, який підтверджує його на складі. Після перевірки наявності товар відправляється через службу доставки до клієнта, а у разі потреби клієнт звертається до сервісного центру за підтримкою [1].

Переваги:

– прямий контакт з клієнтами. Клієнти можуть безпосередньо звертатися до підприємства через додаток, отримуючи швидку відповідь та персоналізовані консультації;

– оптимізація обробки замовлень. Додаток автоматизує процес прийому та обробки замовлень, зменшуючи кількість помилок та підвищуючи швидкість обслуговування;

– збір даних про клієнтів. Застосунок дозволяє зберігати інформацію про клієнтів, що полегшує аналіз їхніх потреб та поведінки.

Недоліки:

– потреба в постійній підтримці додатку. Для ефективної роботи необхідна постійна технічна підтримка та оновлення додатку;

– висока конкуренція. Підприємство повинно забезпечити конкурентні переваги, щоб клієнти обрали саме його додаток [2].

2. Модель продаж через партнерські мережі (Channel Sales Model)

Ця модель передбачає продаж продукції через мережу партнерів, включаючи дилерів та роздрібні магазини (рис. 1.5). Додаток використовується для координації роботи з партнерами, управління замовленнями та обліку запасів.

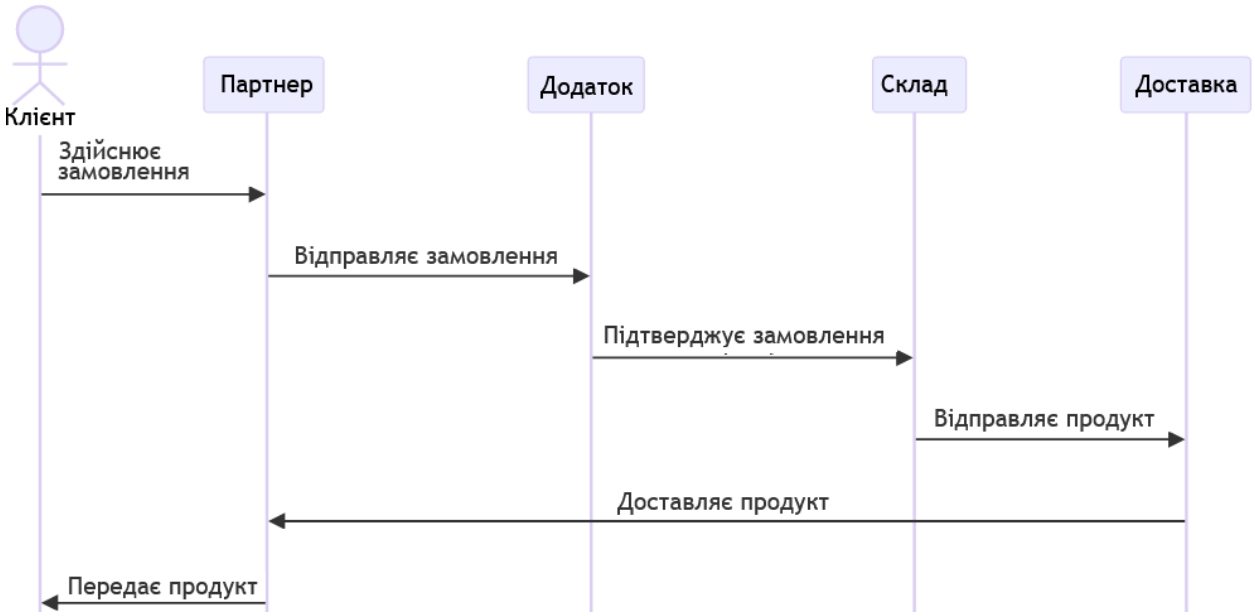


Рисунок 1.5 – Модель продажу через партнерські мережі

У цій моделі клієнт здійснює замовлення у партнера, який використовує додаток для координації та підтвердження замовлення на складі. Після перевірки наявності продукт відправляється зі складу через службу доставки до партнера, який передає його клієнту [3].

Переваги:

– розширення ринку збуту. Співпраця з партнерами дозволяє охопити більшу кількість клієнтів та розширити географію продажів;

– оптимізація роботи з партнерами. Додаток забезпечує ефективну комунікацію з партнерами, полегшує управління замовленнями та відстеження статусу поставок;

– управління запасами. Інтеграція додатку з системою обліку запасів дозволяє оперативно відслідковувати залишки на складах та планувати закупівлі;

Недоліки:

– залежність від партнерів. Успіх продажів залежить від ефективності роботи партнерів, що може створювати додаткові ризики;

– необхідність навчання партнерів. Для ефективного використання додатку необхідно навчити партнерів роботі з ним [4].

3. Модель продаж через електронну комерцію (E-commerce Model)

Модель електронної комерції передбачає продаж продукції через інтернет-магазин, інтегрований з додатком (рис. 1.6). Клієнти можуть переглядати каталог продукції, оформляти замовлення та оплачувати покупки через додаток.

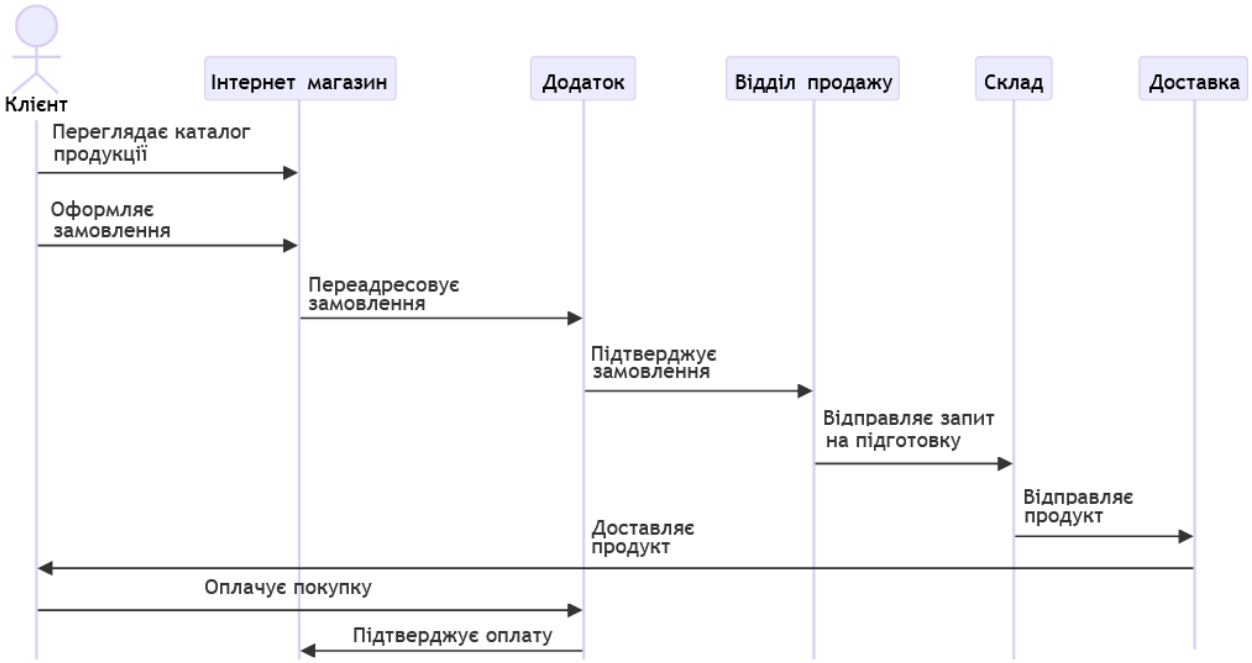


Рисунок 1.6 – Модель продажу через електронну комерцію

У цій моделі клієнт переглядає каталог продукції та оформляє замовлення через інтернет-магазин. Замовлення переадресовується через додаток до відділу продажу, який підтверджує його та відправляє запит на склад для підготовки. Після підготовки продукт доставляється клієнту, який оплачує покупку через додаток [5].

Переваги:

– широка географічна досяжність. Інтернет-магазин дозволяє охопити клієнтів по всій країні та за її межами;

– автоматизація процесів. Додаток автоматизує процеси прийому замовлень, обробки платежів та управління доставкою, зменшуючи операційні витрати та підвищуючи ефективність роботи;

– аналітика та звітність. Інтеграція з додатком дозволяє зберігати та аналізувати дані про замовлення, поведінку клієнтів та ефективність маркетингових кампаній.

Недоліки:

– конкуренція з великими платформами. Підприємство повинно забезпечити конкурентні переваги, щоб клієнти обрали його інтернет-магазин замість великих платформ;

– витрати на просування. Необхідність інвестування в маркетинг та рекламу для залучення клієнтів [6].

Вибір моделі прямих продаж для підприємства «ТехноСервіс» обґрунтований можливістю безпосередньої взаємодії з клієнтами, що дозволяє краще розуміти їхні потреби та надавати персоналізовані консультації. Це сприяє підвищенню рівня задоволеності клієнтів та їх лояльності. Крім того, модель прямих продаж дозволяє оптимізувати процес обробки замовлень за допомогою десктопного додатку, що зменшує кількість помилок та підвищує швидкість обслуговування.

1.4 Аналіз програмних забезпечень для вирішення проблем підприємства

Для ефективного функціонування підприємства «ТехноСервіс» в умовах сучасного ринку необхідне впровадження програмних забезпечень, що дозволять автоматизувати ключові бізнес-процеси та підвищити якість обслуговування клієнтів. Вибір відповідного програмного забезпечення є критичним для оптимізації управління запасами, покращення комунікації з клієнтами та партнерськими мережами, а також для забезпечення точного та своєчасного аналізу даних. У цьому підрозділі будуть розглянуті програмні рішення для визначення їх функціональних можливостей, переваг та

недоліків, що допоможе визначити основний функціонал для розробки власної системи.

Microsoft Dynamics 365 є потужним програмним забезпеченням, яке інтегрує функціональні можливості для управління бізнес-процесами та взаємодії з клієнтами (рис. 1.7) [7]. Призначення цього застосунку полягає в оптимізації та автоматизації процесів управління фінансами, продажами, маркетингом, обслуговуванням клієнтів та операційною діяльністю підприємства.

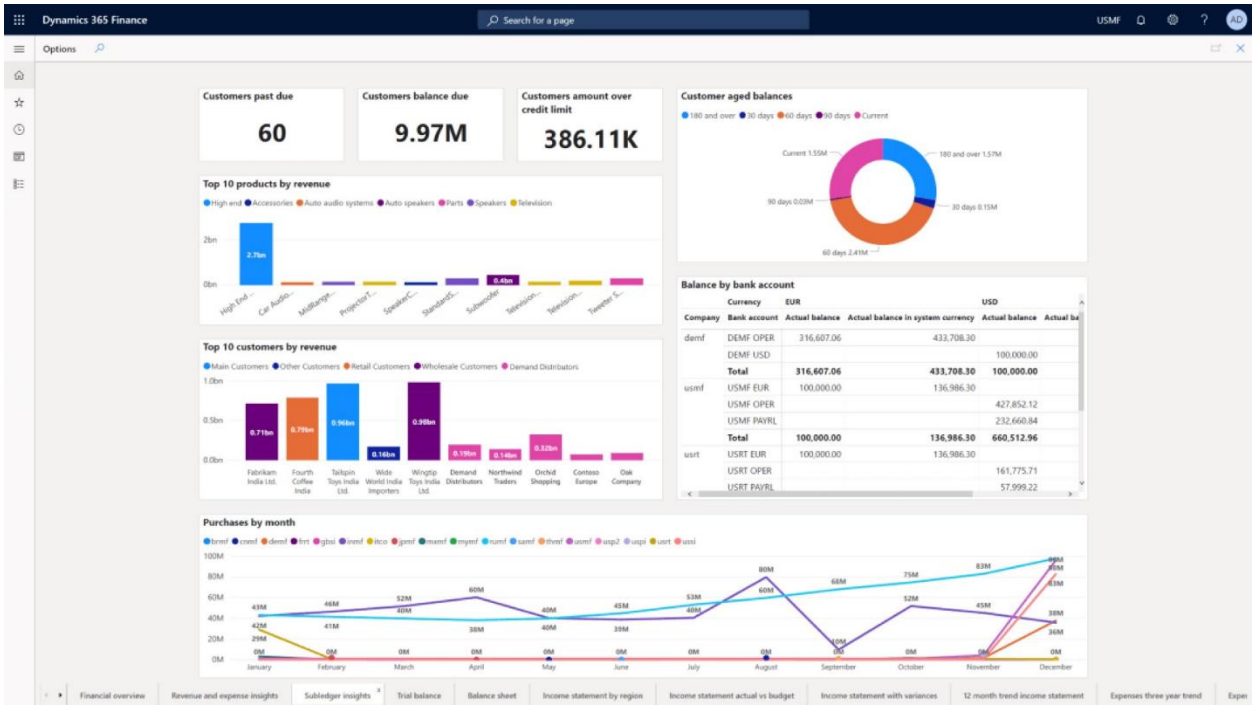


Рисунок 1.7 – Приклад інтерфейсу «Microsoft Dynamics 365»

Архітектура Microsoft Dynamics 365 базується на хмарних технологіях, що забезпечує високу гнучкість, масштабованість та доступність з будь-якого місця. Система складається з окремих модулів, які можуть бути інтегровані між собою для створення комплексного рішення, відповідного до потреб конкретного бізнесу. Використання API та можливостей інтеграції з іншими сервісами дозволяє легко налаштовувати та розширювати функціональність системи [8].

Переваги Microsoft Dynamics 365:

– інтеграція модулів. Забезпечує комплексне рішення для управління всіма аспектами бізнесу, що дозволяє уникнути розрізненості даних;

– хмарна архітектура. Доступність з будь-якого місця, гнучкість і масштабованість;

– можливості налаштування. Висока ступінь налаштування під специфічні потреби бізнесу, що забезпечує гнучкість у використанні;

– аналітика та звітність. Потужні інструменти для аналізу даних та генерації звітів, що допомагають приймати обґрунтовані управлінські рішення.

Недоліки Microsoft Dynamics 365:

– висока вартість. Вартість впровадження та ліцензування може бути досить високою, що робить його менш доступним для малих підприємств;

– складність впровадження. Потребує значних ресурсів і часу для впровадження та налаштування;

– потреба в навчанні. Співробітникам необхідно проходити навчання для ефективного використання системи, що може зайняти певний час і ресурси;

– залежність від Інтернету. Як хмарне рішення, система залежить від надійного інтернет-з'єднання, що може бути проблематичним в умовах нестабільного доступу до мережі [9].

Отже, Microsoft Dynamics 365 є потужним і гнучким інструментом для управління бізнесом, який забезпечує інтеграцію всіх ключових процесів в одній системі. Хмарна архітектура та можливості налаштування роблять його привабливим для великих підприємств, які потребують комплексного рішення для оптимізації своєї діяльності. Проте висока вартість і складність впровадження можуть бути суттєвими перешкодами для менших компаній. Враховуючи ці фактори, Microsoft Dynamics 365 є ефективним рішенням для підприємств, які готові інвестувати в довгостроковий розвиток та автоматизацію своїх бізнес-процесів.

Zoho CRM є програмним забезпеченням для управління взаємодією з клієнтами, призначеним для оптимізації продажів, маркетингу та підтримки клієнтів (рис. 1.8) [10]. Призначення цього застосунку полягає в покращенні

взаємодії з клієнтами, автоматизації бізнес-процесів та підвищенні продуктивності команди продажів.

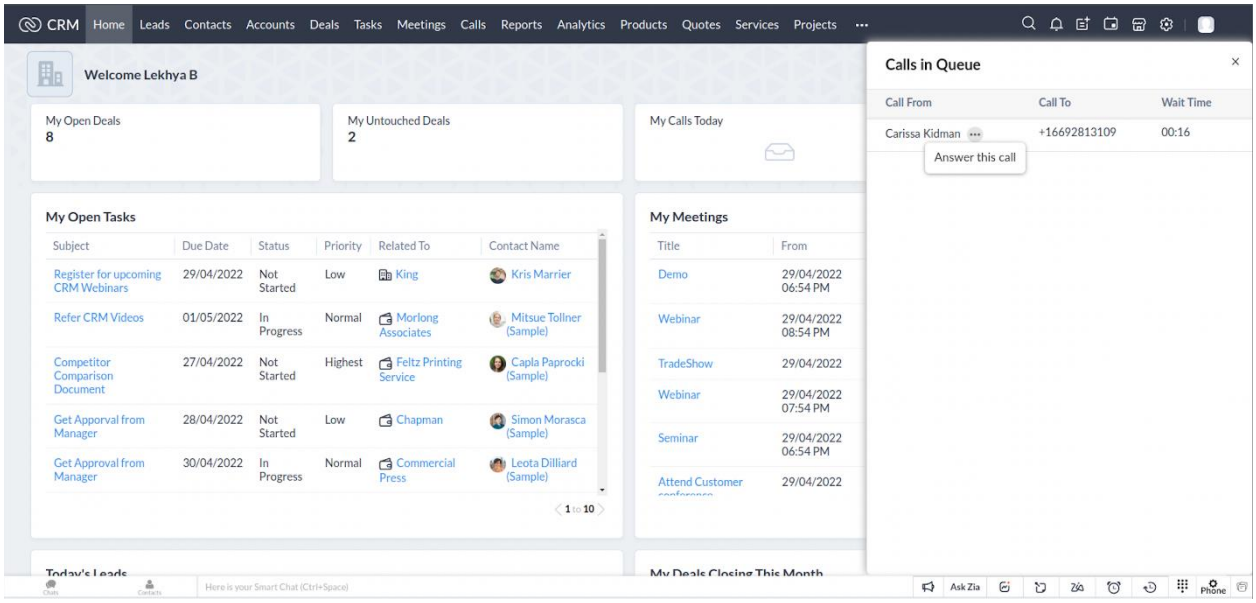


Рисунок 1.8 – Приклад інтерфейсу «Zoho CRM»

Архітектура Zoho CRM базується на хмарних технологіях, що забезпечує доступність з будь-якого місця та на будь-якому пристрої. Система побудована на модульному принципі, що дозволяє легко інтегрувати різні функції, такі як управління контактами, автоматизація маркетингу, аналітика продажів та підтримка клієнтів. Zoho CRM також підтримує інтеграцію з іншими сервісами та додатками через API, що робить його гнучким і легко налаштовуваним під потреби конкретного бізнесу [11].

Переваги Zoho CRM:

– доступна вартість. Zoho CRM пропонує конкурентоспроможну цінову політику, що робить його доступним для малих і середніх підприємств;

– хмарна архітектура. Забезпечує доступність з будь-якого місця та на будь-якому пристрої, що підвищує мобільність користувачів;

– можливості інтеграції. Підтримує інтеграцію з численними сторонніми додатками та сервісами через API, що розширює функціональність системи;

– інтуїтивний інтерфейс. Простий та зручний у використанні інтерфейс, що зменшує потребу в тривалому навчанні персоналу.

Недоліки Zoho CRM:

– обмежена функціональність у базових планах. Деякі розширені функції доступні лише у вищих тарифних планах, що може обмежувати можливості для малих підприємств;

– проблеми з підтримкою. Деякі користувачі відзначають проблеми зі швидкістю та якістю технічної підтримки;

– складність налаштування. Налаштування та адаптація системи під специфічні потреби бізнесу можуть вимагати значних зусиль та часу;

– обмежені можливості кастомізації. Порівняно з іншими CRM-системами, Zoho CRM може мати менші можливості для глибокої кастомізації інтерфейсу та функціоналу [12].

Отже, Zoho CRM є доступним і гнучким інструментом для управління взаємодією з клієнтами, що забезпечує необхідний функціонал для малих та середніх підприємств. Хмарна архітектура, інтеграційні можливості та інтуїтивний інтерфейс роблять його привабливим вибором для бізнесів, що прагнуть підвищити ефективність продажів та маркетингу. Однак, обмеження в базових планах, проблеми з підтримкою та складність налаштування можуть бути викликом для деяких користувачів. Враховуючи ці фактори, Zoho CRM є ефективним рішенням для підприємств, які потребують доступного та зручного інструменту для управління взаємодією з клієнтами.

Odoo є комплексним програмним забезпеченням для управління бізнесом, яке включає модулі для продажів, бухгалтерії, управління запасами, виробництва, проектного менеджменту, маркетингу та багатьох інших аспектів діяльності підприємства (рис. 1.9) [13]. Призначення Odoo полягає в інтеграції всіх бізнес-процесів в єдину систему для підвищення ефективності та прозорості управління.

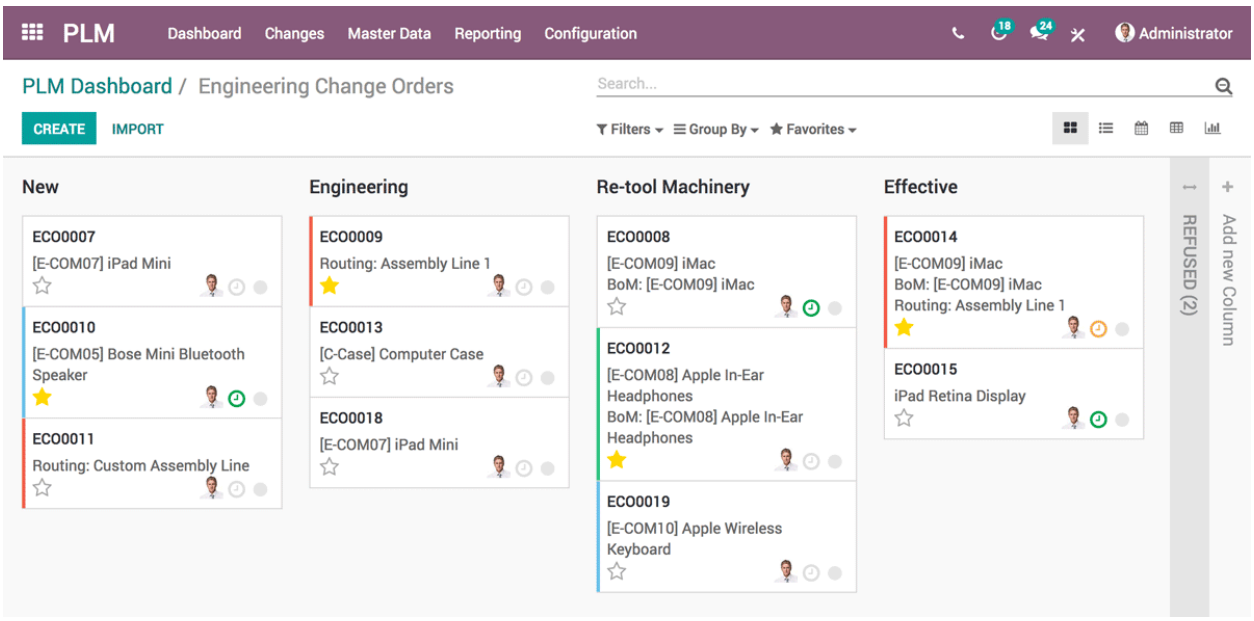


Рисунок 1.9 – Приклад інтерфейсу «Odoo»

Архітектура Odoo базується на модульному принципі, що дозволяє підприємствам вибирати та налаштовувати лише ті модулі, які їм потрібні. Система доступна як у хмарному, так і в локальному варіанті, що забезпечує гнучкість у розгортанні та експлуатації. Odoo підтримує інтеграцію з численними сторонніми сервісами через API, що дозволяє легко розширювати функціональність відповідно до потреб бізнесу [14].

Переваги Odoo:

– модульна архітектура. Можливість вибирати та налаштовувати тільки ті модулі, які потрібні підприємству, що забезпечує гнучкість і економію ресурсів;

– всеосяжний функціонал. Включає широкий спектр модулів для управління всіма аспектами бізнесу, від продажів і маркетингу до бухгалтерії та виробництва;

– доступність як хмарного, так і локального варіанту. Дозволяє підприємствам вибирати найбільш підходящий спосіб розгортання системи відповідно до їх потреб;

– інтеграційні можливості. Підтримує інтеграцію з численними сторонніми сервісами через API, що дозволяє легко розширювати функціональність.

Недоліки Odoo:

– висока складність налаштування. Першопочаткове налаштування та адаптація системи можуть вимагати значних зусиль та часу;

– вартість для великих підприємств. Вартість ліцензій та впровадження може бути високою для великих підприємств з комплексними потребами;

– залежність від сторонньої підтримки. Може знадобитися залучення сторонніх консультантів для впровадження та налаштування, що збільшує загальні витрати;

– обмежена функціональність у безкоштовній версії. Деякі розширені функції доступні тільки у платних версіях, що може обмежувати можливості для малих підприємств [15].

Отже, Odoo є потужною і гнучкою системою управління бізнесом, яка підходить для широкого спектру підприємств завдяки своїй модульній архітектурі та всеосяжному функціоналу. Можливість вибору між хмарним і локальним варіантом розгортання, а також широкі інтеграційні можливості роблять Odoo привабливим вибором для компаній, що прагнуть інтегрувати всі свої бізнес-процеси в одну систему. Однак, висока складність налаштування, вартість для великих підприємств і залежність від сторонньої підтримки можуть бути значними викликами для деяких організацій. Враховуючи ці фактори, Odoo є ефективним рішенням для підприємств, які потребують комплексного та гнучкого інструменту для управління своєю діяльністю.

На основі проведеного аналізу програмних забезпечень, стає очевидним, що існуючі рішення, такі як Microsoft Dynamics 365, Zoho CRM та Odoo, мають високу вартість, складність налаштування та можуть не повністю відповідати специфічним потребам підприємства «ТехноСервіс». Розробка власного ПЗ дозволить створити оптимізовану систему, яка буде враховувати унікальні бізнес-процеси та вимоги підприємства, забезпечуючи ефективну інтеграцію та управління всіма аспектами його діяльності без зайвих витрат на непотрібні функції.

1.5 Постановка задачі дослідження

З огляду на виявлені потреби підприємства «ТехноСервіс» та аналіз існуючих програмних рішень, необхідно розробити власне програмне забезпечення, яке б враховувало специфічні бізнес-процеси компанії та забезпечувало інтегроване управління всіма аспектами її діяльності.

Для досягнення цієї мети необхідно розробити наступний функціонал застосунку:

– розробити систему управління, що включатиме модулі для купівлі, підтвердження купівлі, замовлення товару, надходження, а також облік ремонту техніки;

– створити довідники для ефективного управління категоріями комплектуючих, продукцією, постачальниками та умовами постачання;

– розробити систему звітності, яка забезпечить генерацію звітів про продаж та надходження товарів за обраний період часу;

– впровадити модулі системного адміністрування для управління користувачами системи, подіями, персоналізацією та зміною користувача.

Кожен з цих модулів необхідно розробити з урахуванням специфіки ролей користувачів:

– директор. Веде дані про асортимент товару, постачальників та умови постачання товару;

– покупець. Переглядає асортимент та оформлює замовлення на покупку товару;

– продавець. Обробляє всі замовлення, перевіряє наявність товару на складі, оформлює накладні на постачання або продаж товару;

– робітник. Веде облік постачання та продажу товару, а також облік ремонту техніки.

Розробка власного програмного забезпечення дозволить підприємству «ТехноСервіс» інтегрувати всі бізнес-процеси в єдину систему, що

забезпечить оптимізацію управління, підвищення ефективності обслуговування клієнтів та конкурентоспроможність на ринку.

1.6 Висновок

У рамках даного розділу було проведено всебічний аналіз організації, функціонування підприємства «ТехноСервіс» та дослідження існуючих моделей продаж і взаємодії з клієнтами. Було розглянуто організаційну структуру підприємства, визначено основні принципи його роботи, включаючи клієнтоорієнтованість, ефективність процесів, інноваційність, безперервний розвиток персоналу, взаємодію та комунікацію. Детально проаналізовано проблеми, з якими зіштовхується «ТехноСервіс» у сучасних умовах, такі як управління запасами, автоматизація бізнес-процесів, забезпечення якості обслуговування та інтеграція з існуючими системами.

Було здійснено огляд існуючих моделей продаж, включаючи модель прямих продаж, модель продаж через партнерські мережі та модель продажу через електронну комерцію. На основі цього аналізу було обрано модель прямих продаж для реалізації, як найбільш відповідну для підприємства. Також проведено аналіз програмних забезпечень, таких як Microsoft Dynamics 365, Zoho CRM та Odoo, де виділено їх переваги та недоліки. На основі цього аналізу прийнято рішення про необхідність розробки власної системи для «ТехноСервіс», оскільки існуючі рішення мають високу вартість, складність налаштування та не завжди відповідають специфічним потребам підприємства.

Було поставлено конкретні задачі дослідження, включаючи розробку функціоналу системи управління, створення довідників, розробку системи звітності та модулів системного адміністрування. Визначено ролі користувачів системи: директора, покупця, продавця та робітника, що дозволяє чітко окреслити обов'язки та взаємодії кожної ролі в майбутній системі. Результати цього розділу надають чітке розуміння напрямків, які будуть розвиватися в наступному розділі, включаючи детальну розробку системи, її

функціональних можливостей та інтеграційних рішень для досягнення оптимальної ефективності та продуктивності підприємства «ТехноСервіс».

2 ВИБІР ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір технологій для реалізації проекту

При розробці власної інформаційної системи для підприємства «ТехноСервіс» важливим етапом є вибір відповідної мови програмування. Вибір мови залежить від багатьох факторів, включаючи вимоги до продуктивності, легкість у розробці, можливості інтеграції з іншими системами та доступність фахівців на ринку праці. Для нашого аналізу ми розглянемо три популярні мови програмування: Python, Java та C#.

Python є високорівневою мовою програмування, відомою своєю простотою та легкістю у вивченні. Вона широко використовується для розробки веб-додатків, наукових досліджень, аналізу даних та машинного навчання. Python підтримує велику кількість бібліотек і фреймворків, що робить її універсальним інструментом для розробки різних типів додатків [16].

Java є об'єктно-орієнтованою мовою програмування, яка забезпечує високу продуктивність та кросплатформеність завдяки своїй віртуальній машині JVM (Java Virtual Machine). Вона широко використовується для розробки корпоративних додатків, мобільних додатків (особливо на платформі Android) та веб-додатків. Java відома своєю стабільністю та масштабованістю, що робить її популярним вибором для великих проектів [17].

C# є об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft, яка є основною мовою для платформи .NET. Вона широко використовується для розробки десктопних додатків, веб-додатків та ігор (зокрема за допомогою Unity). C# забезпечує високу інтеграцію з продуктами Microsoft та підтримує широкий спектр функцій для розробки сучасних додатків [18].

У табл. 2.1 представлено сильні та слабкі сторони, які необхідно враховувати при виборі інструменту для розробки інформаційної системи.

Таблиця 2.1 – Порівняльний аналіз мов програмування

Характеристика	Python	Java	C#
Легкість у вивченні	Дуже легка, проста синтаксис	Помірна складність, строгий синтаксис	Помірна складність, схожа на Java
Продуктивність	Відносно низька, інтерпретована	Висока, компілюється до байт-коду	Висока, компілюється до машинного коду
Можливості інтеграції	Високі, багато бібліотек та фреймворків	Високі, багато бібліотек та фреймворків	Дуже високі, особливо з продуктами Microsoft
Використання в корпораціях	Часто використовується для скриптів, аналізу даних	Дуже популярна для великих корпоративних проектів	Дуже популярна для корпоративних додатків на платформі .NET
Підтримка IDE	Відмінна (PyCharm, VS Code)	Відмінна (IntelliJ IDEA, Eclipse)	Відмінна (Visual Studio, Rider)
Безпека	Висока, але залежить від бібліотек	Висока, багато вбудованих засобів безпеки	Дуже висока, інтеграція з безпековими функціями .NET
Спільнота та підтримка	Дуже велика та активна спільнота	Дуже велика та активна спільнота	Велика та активна спільнота, особливо серед користувачів Microsoft

Вибір мови програмування C# для розробки системи «ТехноСервіс» обґрунтований її високою продуктивністю та можливістю компіляції до машинного коду, що забезпечує швидкодію та ефективність роботи додатку. C# відома своїми потужними інтеграційними можливостями, особливо з продуктами Microsoft, що дозволяє легко інтегрувати нову систему з існуючими корпоративними рішеннями. Крім того, C# є популярною мовою для розробки корпоративних додатків на платформі .NET, що забезпечує

високу стабільність та масштабованість проекту. Високий рівень безпеки, забезпечений інтеграцією з безпековими функціями .NET, робить C# ідеальним вибором для створення надійного та захищеного програмного забезпечення.

При розробці інформаційної системи для підприємства «ТехноСервіс» важливим етапом є вибір системи управління базами даних (СУБД). Вибір СУБД залежить від вимог до продуктивності, масштабованості, надійності та вартості володіння. Для нашого аналізу ми розглянемо три популярні СУБД: MS SQL Server, MySQL та Oracle.

MS SQL Server є потужною реляційною СУБД, розробленою компанією Microsoft. Вона відома своєю високою продуктивністю, масштабованістю та надійністю, що робить її популярним вибором для корпоративних додатків. MS SQL Server забезпечує тісну інтеграцію з іншими продуктами Microsoft, такими як .NET, що спрощує розробку та управління базами даних у середовищі Windows [19-20].

MySQL є однією з найпопулярніших відкритих реляційних СУБД, відомою своєю швидкістю та простотою використання. Вона широко використовується для веб-додатків та пропонує високий рівень продуктивності навіть при великих обсягах даних. MySQL підтримує численні платформи та забезпечує високу гнучкість у виборі операційних систем, що робить її популярним вибором для багатьох розробників [21].

Oracle Database є однією з найпотужніших і найбільш функціональних реляційних СУБД на ринку. Вона забезпечує високу продуктивність, масштабованість та надійність, що робить її ідеальним вибором для великих корпоративних систем. Oracle пропонує широкий спектр інструментів для управління базами даних та аналітики, що дозволяє ефективно обробляти великі обсяги даних та складні запити [22].

Кожна з цих СУБД має свої унікальні переваги та недоліки, які необхідно враховувати при виборі інструменту для розробки інформаційної системи (табл. 2.2).

Таблиця 2.2 – Порівняльний аналіз СУБД

Характеристика	MS SQL Server	MySQL	Oracle
Ліцензія	Комерційна, платна	Відкрита (GPL), безкоштовна	Комерційна, платна
Продуктивність	Висока, особливо на Windows	Висока, оптимізована для веб	Дуже висока, особливо для великих обсягів даних
Масштабованість	Висока	Висока	Дуже висока
Надійність	Дуже висока	Висока	Дуже висока
Підтримка платформ	Windows	Багатоплатформна (Windows, Linux, Unix)	Багатоплатформна (Windows, Linux, Unix)
Інтеграція	Тісна інтеграція з продуктами Microsoft	Широка підтримка через API	Широка підтримка через API
Гнучкість налаштування	Висока	Висока	Дуже висока
Спільнота та підтримка	Відмінна підтримка від Microsoft	Велика спільнота, активна підтримка	Відмінна підтримка від Oracle
Вартість володіння	Висока	Низька	Дуже висока

Вибір MySQL для розробки інформаційної системи підприємства «ТехноСервіс» обґрунтований її відкритою ліцензією, що значно знижує загальні витрати на впровадження та експлуатацію системи. MySQL забезпечує високу продуктивність та гнучкість налаштування, що дозволяє оптимізувати базу даних під специфічні потреби підприємства. Багатоплатформна підтримка робить MySQL універсальним рішенням, яке можна легко інтегрувати з існуючою інфраструктурою. Завдяки великій та активній спільноті користувачів, MySQL пропонує широкий спектр ресурсів та підтримки, що спрощує процес розробки та обслуговування системи.

2.2 Визначення вимог до системи. Побудова діаграм прецедентів

При розробці інформаційної системи для підприємства «ТехноСервіс» важливим етапом є визначення вимог до системи. Це дозволяє чітко окреслити функціональні можливості та забезпечити відповідність розробки специфічним потребам підприємства. Вимоги до системи визначаються на основі аналізу бізнес-процесів, потреб користувачів та технічних умов експлуатації. Вони включають функціональні та нефункціональні аспекти, що забезпечують надійну роботу системи та задоволення очікувань користувачів. Для кращого розуміння функціональних можливостей системи, у табл. 2.3 наведено основні функціональні вимоги до застосунку.

Таблиця 2.3 – Функціональні вимоги системи

Вимоги	Опис
REQ-1	Система повинна дозволяти користувачам створювати, редагувати та підтверджувати купівлю товарів, а також відслідковувати статус замовлень
REQ-2	Застосунок повинен забезпечувати можливість замовлення товарів у постачальників, включаючи автоматичне створення замовлень на основі аналізу запасів
REQ-3	Система повинна дозволяти ведення обліку ремонтних робіт, включаючи реєстрацію заявок на ремонт, обробку та відстеження виконання ремонтних робіт
REQ-4	Застосунок повинен мати модулі для управління категоріями комплектуючих, продукцією, постачальниками та умовами постачання, що дозволить ефективно керувати даними
REQ-5	Система повинна надавати функції для генерації звітів про продажі та надходження товарів за обраний період часу, забезпечуючи аналітичну підтримку для прийняття управлінських рішень
REQ-6	Застосунок повинен включати функції для управління користувачами системи, ведення журналу подій, що забезпечить безпеку системи

Нефункціональні вимоги мають важливе значення для забезпечення якості кінцевого програмного продукту. Ці вимоги стосуються аспектів системи, які не є її основними функціями, але впливають на ефективність та

зручність використання. Вони критично важливі, оскільки забезпечують стабільність системи та відповідають очікуванням користувачів, як показано в табл. 2.4.

Таблиця 2.4 – Нефункціональні вимоги системи

Вимоги	Опис
REQ-7	Система повинна забезпечувати високу швидкість обробки запитів та виконання операцій, забезпечуючи мінімальний час відгуку навіть при високому навантаженні.
REQ-8	Застосунок повинен мати високу надійність та забезпечувати безперебійну роботу, з мінімальним часом простою, використовуючи механізми резервного копіювання та відновлення даних
REQ-9	Система повинна забезпечувати високий рівень безпеки даних, включаючи захист від несанкціонованого доступу, шифрування даних та управління правами доступу користувачів
REQ-10	Застосунок повинен бути здатний до масштабування, щоб ефективно обробляти зростаючі обсяги даних та кількість користувачів без значного зниження продуктивності
REQ-11	Інтерфейс користувача повинен бути інтуїтивно зрозумілим, простим у використанні та забезпечувати зручний доступ до основних функцій системи для всіх категорій користувачів

У розробленій інформаційній системі для підприємства «ТехноСервіс» передбачено кілька ролей користувачів, кожна з яких має свої специфічні функції та обов'язки. Це дозволяє ефективно розподілити завдання між працівниками, забезпечуючи оптимальне управління та обслуговування клієнтів. Кожна роль користувача визначає рівень доступу до різних функцій системи та відповідальність за виконання певних бізнес-процесів (рис. 2.5). Це стимулює адаптацію системи під вимоги користувачів, ефективно забезпечуючи задоволення потреб та досягнення необхідних цілей акторами системи.

Таблиця 2.5 – Ролі користувачів та їхні цілі у системі

Актори	Цілі
Директор	Директор відповідає за ведення даних про асортимент товару, постачальників та умови постачання товару. Він має доступ до управління довідниками, контролює закупівлі та оцінює ефективність роботи підприємства. Директор також використовує систему для генерації звітів та аналізу даних, що допомагає приймати стратегічні рішення.
Робітник	Покупець має можливість переглядати асортимент продукції через систему та оформлювати замовлення на покупку товару. Він взаємодіє з системою для пошуку необхідних товарів, додавання їх до кошика та підтвердження замовлення. Покупець отримує інформацію про статус замовлення та може слідкувати за його виконанням.
Продавець	Продавець обробляє всі замовлення, перевіряє наявність товару на складі та оформлює накладні на постачання або продаж товару. Він відповідає за своєчасне виконання замовлень, комунікацію з клієнтами та координацію з постачальниками. Продавець також оновлює інформацію про наявність товарів у системі та відстежує їх рух.
Покупець	Робітник веде облік постачання та продажу товару, а також облік ремонту техніки. Він реєструє надходження товарів на склад, фіксує їх продажі та забезпечує точність даних про залишки. Крім того, робітник обробляє заявки на ремонт техніки, контролює виконання ремонтних робіт та оновлює відповідну інформацію у системі.

Варіанти використання (use cases) для інформаційної системи підприємства «ТехноСервіс» описують різні сценарії взаємодії користувачів із системою для виконання конкретних бізнес-завдань [23]. Вони охоплюють основні операції, такі як управління купівлею та замовленнями товарів, облік ремонту, а також генерацію звітів. Кожен варіант використання деталізує послідовність дій користувача та відповідає визначеним ролям, що забезпечує ефективну роботу системи та досягнення бізнес-цілей.

У табл. 2.6 представлено основні варіанти використання для системи «ТехноСервіс», включаючи їхні описи та відповідні ролі користувачів.

Таблиця 2.6 – Опис варіантів використання системи

Id	Ім’я	Опис
1	2	3
UC1	Реєстрація користувачів	Процес створення нового користувача у системі.
UC2	Авторизація користувачів	Вхід користувача в систему за допомогою облікових даних.
UC3	Перегляд зареєстрованих користувачів	Відображення списку всіх зареєстрованих користувачів системи.
UC4	Додавання користувача	Створення нового користувача з визначеними ролями та правами доступу.
UC5	Редагування користувача	Зміна даних зареєстрованого користувача.
UC6	Перегляд категорій	Відображення списку категорій комплектуючих.
UC7	Додавання категорії	Додавання нової категорії комплектуючих у систему.
UC8	Редагування категорії	Зміна даних існуючої категорії комплектуючих.
UC9	Перегляд продукції	Відображення списку наявної продукції.
UC10	Додавання продукції	Додавання нової продукції у систему.
UC11	Редагування продукції	Зміна даних існуючої продукції.
UC12	Формування накладної	Створення накладної на продаж або постачання товарів.
UC13	Підтвердження видачі продукції	Підтвердження факту видачі продукції клієнту.
UC14	Накладна для постачання	Створення накладної для отримання товарів від постачальників.
UC15	Підтвердження надходження	Підтвердження факту надходження товарів на склад.

Продовження табл. 2.6

UC16	Перегляд постачальників	Відображення списку постачальників товарів.
UC17	Додавання постачальника	Додавання нового постачальника у систему.

UC18	Редагування постачальника	Зміна даних існуючого постачальника.
UC19	Перегляд умов доставки	Відображення умов доставки товарів.
UC20	Додавання умови доставки	Додавання нових умов доставки у систему.
UC21	Зміна умови доставки	Зміна існуючих умов доставки товарів.
UC22	Огляд ремонтної техніки	Перегляд інформації про техніку, що потребує ремонту.
UC23	Додавання нового ремонту	Реєстрація нової заявки на ремонт техніки.
UC24	Оновлення даних ремонту	Зміна даних про поточні ремонтні роботи.
UC25	Звітність про продаж продукції	Генерація звітів про продаж продукції за обраний період.
UC26	Звітність по надходженню	Генерація звітів про надходження товарів за обраний період.
UC27	Перегляд подій системи	Відображення журналу подій системи для моніторингу та аудиту.

На основі вимог до системи розроблено use-case діаграми прецедентів для всіх ролей, зокрема: директор, робітник, продавець та покупець. Діаграми варіантів використання системи представлені на рис. 2.1- 2.4 відповідно.

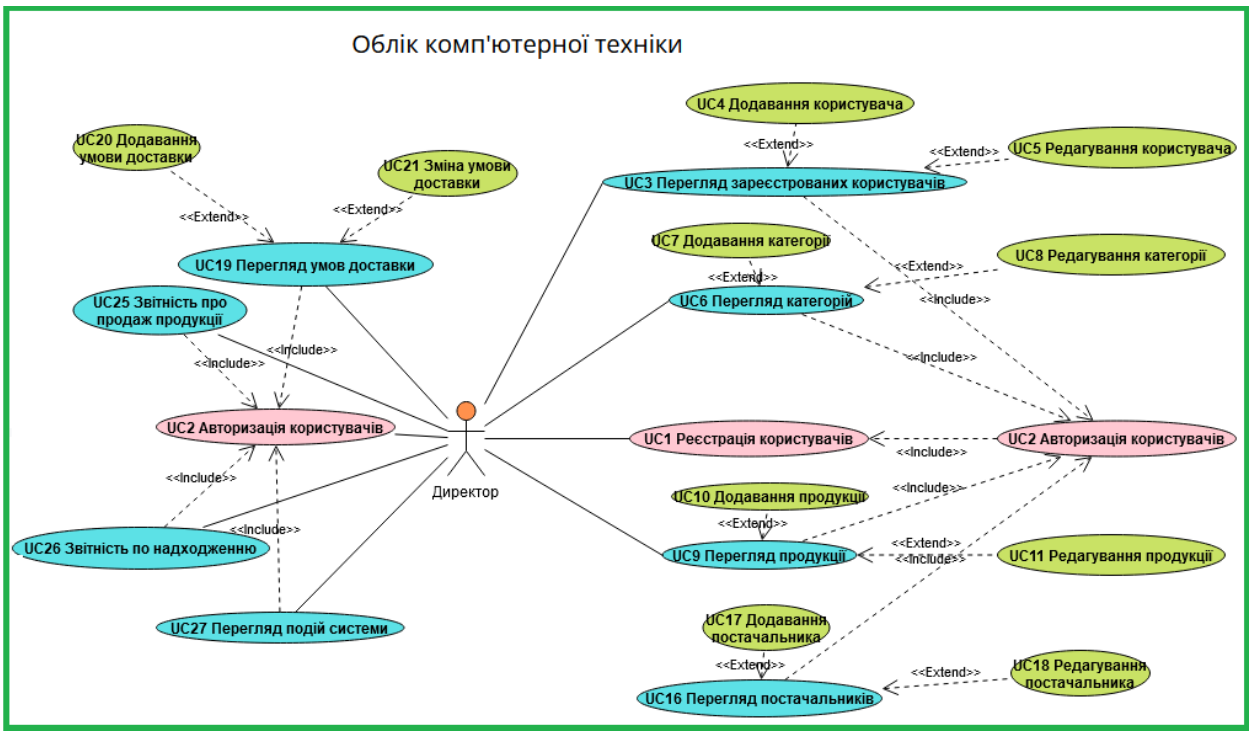


Рисунок 2.1 – Діаграма прецедентів для ролі директора

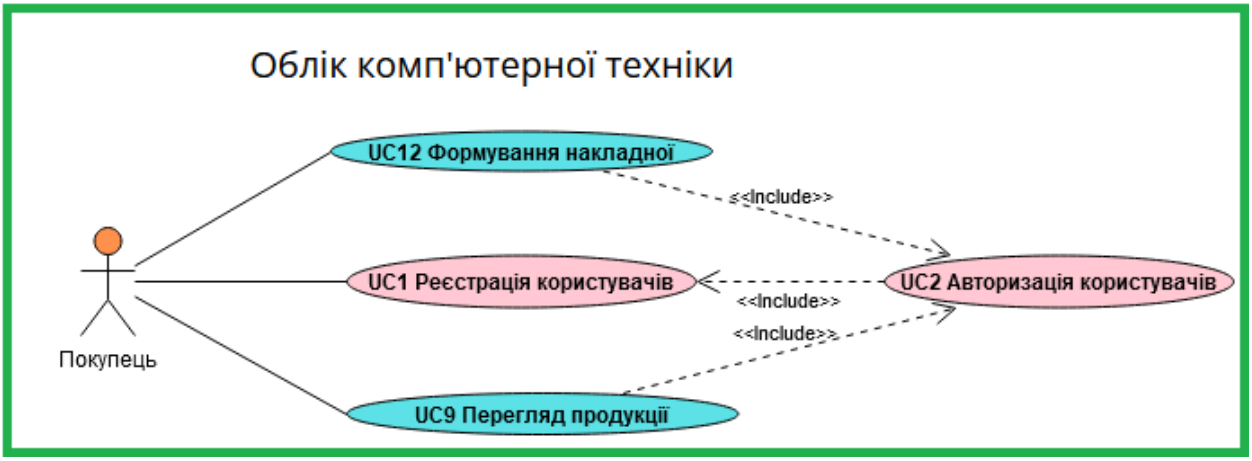


Рисунок 2.2 – Діаграма прецедентів для ролі покупця

Рисунок 2.3 – Діаграма прецедентів для ролі робітника

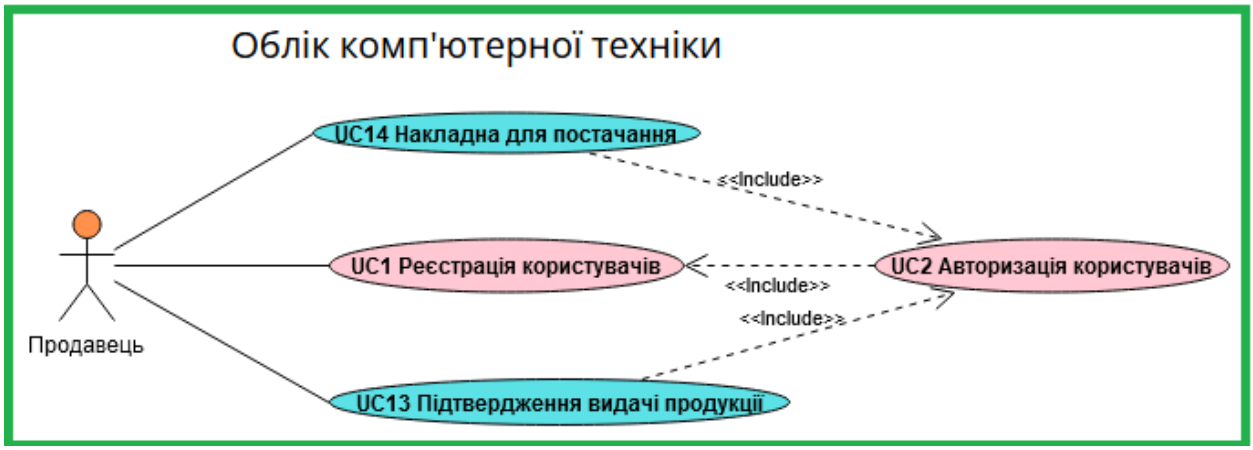


Рисунок 2.4 – Діаграма прецедентів ролі продавця

Після побудови діаграм прецедентів для кожної ролі системи «ТехноСервіс» отримано візуальне представлення всіх можливих взаємодій користувачів із системою. Це дозволяє чітко зрозуміти функціональні вимоги та сценарії використання, забезпечуючи основу для подальшої розробки та тестування програмного забезпечення. Такий підхід допомагає визначити основні завдання для кожної ролі та оптимізувати бізнес-процеси підприємства.

2.3 Моделювання основних бізнес-процесів

Для ефективної розробки інформаційної системи «ТехноСервіс» необхідно здійснити моделювання основних бізнес-процесів, які забезпечують її функціонування. Це дозволяє виявити ключові етапи роботи та оптимізувати їх, що сприяє підвищенню загальної ефективності підприємства. Моделювання включає візуалізацію процесів, таких як купівля, замовлення товарів, облік ремонту, а також управління постачанням продукції. У результаті отримано чітке уявлення про потоки даних і взаємодії між різними ролями користувачів.

На рис. 2.5 зображено процес додавання умов постачання продукції на підприємство, що дозволяє деталізувати функціональні вимоги до застосунку та забезпечити їх ефективну реалізацію.

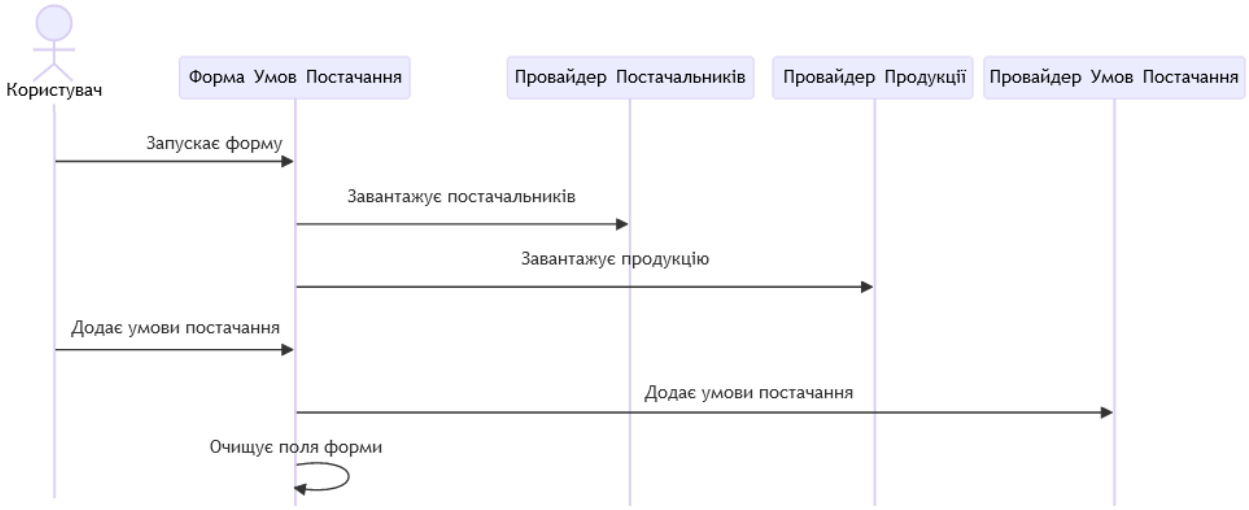


Рисунок 2.5 – Процес додавання умов постачання продукції

По початку процесу користувач запускає форму умов постачання, яка завантажує дані про постачальників і продукцію. Користувач додає умови постачання, після чого форма передає ці дані провайдеру умов постачання та очищує поля форми для наступного вводу.

Рис. 2.6 ілюструє діаграму послідовності для процесу формування накладної для купівлі комплектуючих.

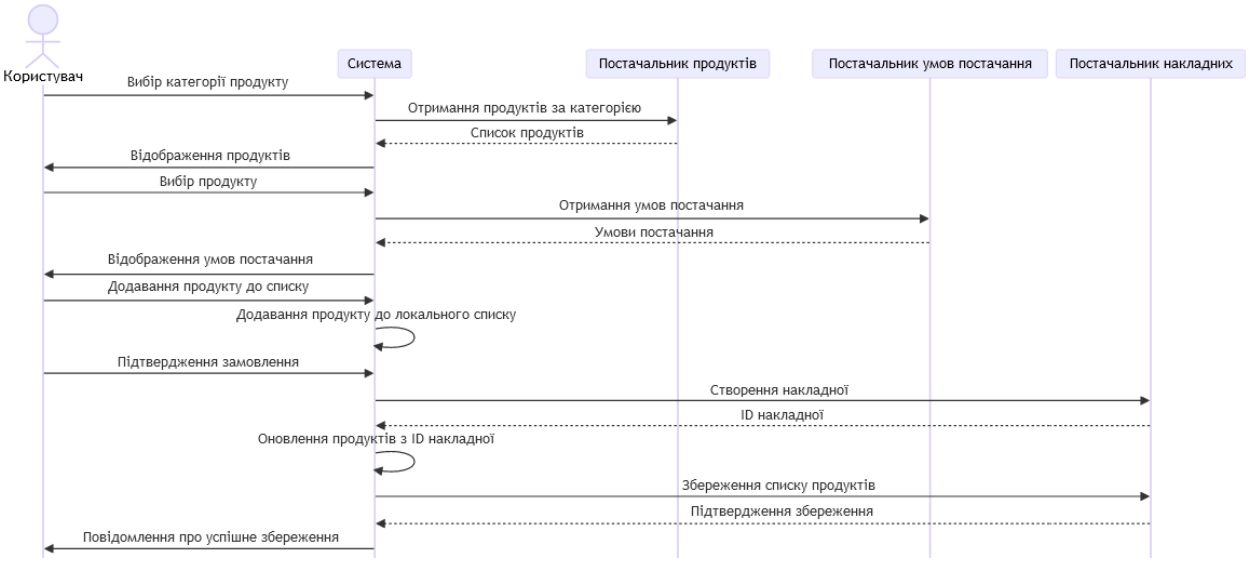


Рисунок 2.6 – Процес формування накладної для купівлі комплектуючих

Користувач обирає категорію продукту, після чого система запитує список продуктів у постачальника продуктів та відображає його користувачу. Далі користувач обирає конкретний продукт, і система отримує умови постачання від постачальника умов постачання та відображає їх. Користувач додає продукт до списку, і система оновлює локальний список продуктів.

Після підтвердження замовлення система створює накладну, зберігає список продуктів у постачальника накладних та повідомляє користувача про успішне збереження замовлення.

На рис. 2.7 представлено діаграму послідовності, яка ілюструє процес формування звіту по надходженню продукції на підприємство.

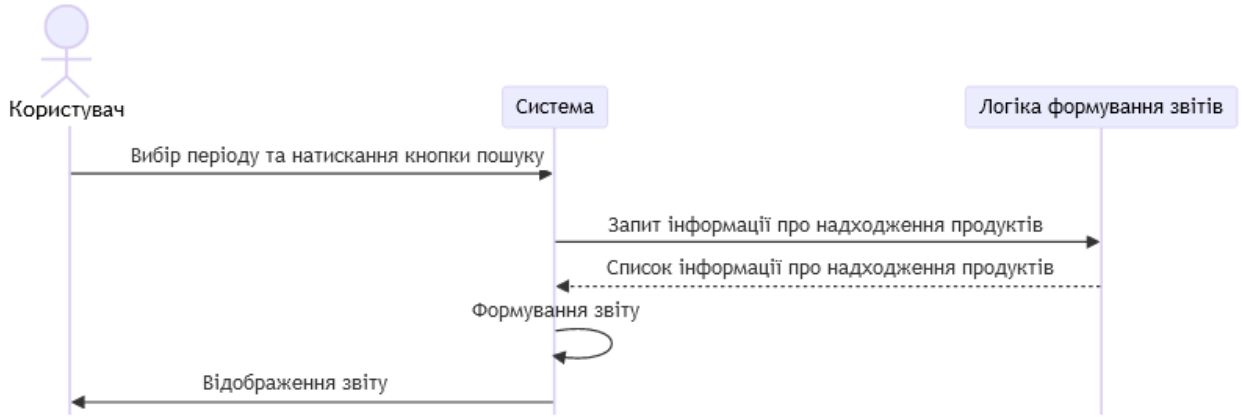


Рисунок 2.7 – Процес формування звітності

Користувач обирає період та натискає кнопку пошуку, після чого система запитує інформацію про надходження продуктів у логіки формування звітів. Логіка формування звітів повертає список інформації про надходження продуктів. Система формує звіт на основі отриманої інформації. Після цього звіт відображається користувачу.

2.4 Проєктування та розробка структури бази даних

Проєктування та розробка структури бази даних є критично важливим етапом у створенні інформаційної системи «ТехноСервіс». Цей процес включає визначення всіх необхідних таблиць, їх атрибутів, взаємозв'язків та обмежень, що забезпечують цілісність і узгодженість даних. Правильно спроєктована база даних дозволяє ефективно зберігати, обробляти та отримувати інформацію, що сприяє оптимізації бізнес-процесів підприємства. Основною метою цього етапу є створення такої структури бази даних, яка б відповідала всім функціональним вимогам системи та забезпечувала високу продуктивність і масштабованість.

База даних системи включає наступні таблиці:

Таблиця «ServiceRequests» призначена для зберігання інформації про запити на обслуговування, зроблені користувачами. Вона є важливою для управління та моніторингу ремонтних робіт, оцінки витрат та відстеження статусу обслуговування пристроїв.

Таблиця 2.7 – Атрибути сутності «ServiceRequests»

№	Поле	Тип даних	Призначення поля
1	ServiceRequestsId	INT	Унікальний ідентифікатор запиту на обслуговування
2	UsersId	INT	Ідентифікатор користувача, який створив запит
3	DeviceName	VARCHAR(150)	Назва пристрою, що потребує обслуговування
4	ServiceDate	DATE	Дата обслуговування
5	ServiceType	VARCHAR(45)	Тип обслуговування
6	Cost	DECIMAL(12,2)	Вартість обслуговування
7	Description	TEXT	Опис проблеми або необхідних робіт
8	Status	TINYINT(1)	Статус запиту на обслуговування (наприклад, виконано/не виконано)

Таблиця «OrderList» призначена для зберігання інформації про позиції замовлення, включаючи продукцію, кількість та ціну продажу. Вона є важливою для деталізації замовлень та відстеження складу кожного замовлення.

Таблиця 2.8 – Атрибути сутності «OrderList»

№	Поле	Тип даних	Призначення поля
1	OrderListId	INT	Унікальний ідентифікатор позиції замовлення
2	OrderId	INT	Ідентифікатор замовлення
3	ProductId	INT	Ідентифікатор продукції
4	Quantity	INT	Кількість продукції в замовленні
5	SalePrice	DECIMAL(10,2)	Ціна продажу продукції

Таблиця «Products» призначена для зберігання інформації про продукцію, доступну для продажу в «ТехноСервіс». Вона є важливою для управління асортиментом товарів, включаючи дані про назву, модель, виробника, ціну продажу, кількість та опис продукції.

Таблиця 2.9 – Атрибути сутності «Products»

№	Поле	Тип даних	Призначення поля
1	ProductId	INT	Унікальний ідентифікатор продукції
2	ProductName	VARCHAR(255)	Назва продукції
3	Model	VARCHAR(255)	Модель продукції
4	Manufacturer	VARCHAR(255)	Виробник продукції
5	SalePrice	DECIMAL(12,2)	Ціна продажу продукції
6	Description	TEXT	Опис продукції
7	Quantity	INT	Кількість доступної продукції
8	CategoryId	INT	Ідентифікатор категорії, до якої належить продукція

Таблиця «SalesInvoices» призначена для зберігання інформації про накладні на продаж, включаючи дату, загальну вартість, постачальника та статус накладної. Вона є важливою для управління процесом продажу, обліку взаємодії з постачальниками та аналізу фінансових операцій.

Таблиця 2.10 – Атрибути сутності «SalesInvoices»

№	Поле	Тип даних	Призначення поля
1	SalesInvoiceId	INT	Унікальний ідентифікатор накладної на продаж
2	UsersId	INT	Ідентифікатор користувача, який створив накладну
3	InvoiceDate	DATE	Дата створення накладної
4	TotalPrice	DECIMAL(10,2)	Загальна вартість накладної
5	SupplierId	INT	Ідентифікатор постачальника
6	Status	TINYINT(1)	Статус накладної (наприклад, виконано/не виконано)

Таблиця «Category» призначена для зберігання інформації про категорії комплектуючих. Вона є важливою для організації та управління каталогом продукції, що полегшує пошук та класифікацію товарів.

Таблиця 2.11 – Атрибути сутності «ServiceRequests»

№	Поле	Тип даних	Призначення поля
1	CategoryId	INT	Унікальний ідентифікатор категорії
2	CategoryName	VARCHAR(100)	Назва категорії
3	Description	TEXT	Опис категорії, що містить додаткову інформацію

Таблиця «Logs» призначена для зберігання записів про події та дії, виконані користувачами в системі. Вона є важливою для моніторингу активності, аудиту безпеки та аналізу поведінки користувачів.

Таблиця 2.12 – Атрибути сутності «Logs»

№	Поле	Тип даних	Призначення поля
1	LogsId	INT	Унікальний ідентифікатор журналу подій
2	UsersId	INT	Ідентифікатор користувача події
3	EventNameShow	TEXT	Назва події
4	EventDate	DATETIME	Дата та час події

Таблиця «Orders» призначена для зберігання інформації про замовлення, зроблені користувачами, включаючи дату замовлення, загальну вартість та статус замовлення. Вона є важливою для управління процесом замовлень та обліку взаємодії з клієнтами.

Таблиця 2.13 – Атрибути сутності «Orders»

№	Поле	Тип даних	Призначення поля
1	OrderId	INT	Унікальний ідентифікатор замовлення
2	UsersId	INT	Ідентифікатор користувача замовлення
3	OrderDate	DATE	Дата створення замовлення
4	TotalPrice	DECIMAL(12,2)	Загальна вартість замовлення
5	Status	TINYINT(1)	Статус замовлення (виконано/не виконано)

Таблиця «ProductsList» призначена для зберігання інформації про товари, включені до накладних на продаж. Вона є важливою для деталізації накладних, відстеження кількості проданих товарів та їхніх цін закупівлі.

Таблиця 2.14 – Атрибути сутності «Products»

№	Поле	Тип даних	Призначення поля
1	ProductsListId	INT	Унікальний ідентифікатор позиції у накладній
2	SalesInvoicesId	INT	Ідентифікатор накладної на продаж
3	ProductId	INT	Ідентифікатор продукції
4	Quantity	INT	Кількість проданих товарів
5	PurchasePrice	DECIMAL(10,2)	Ціна закупівлі товару

Таблиця «SupplyConditions» призначена для зберігання інформації про умови постачання товарів від різних постачальників. Вона є важливою для управління деталями постачання, включаючи умови доставки, мінімальні та максимальні обсяги, а також ціну за одиницю продукції.

Таблиця 2.7 – Атрибути сутності «SupplyConditions»

№	Поле	Тип даних	Призначення поля
1	SupplyConditionId	INT	Унікальний ідентифікатор умови постачання
2	SupplierId	INT	Ідентифікатор постачальника
3	ProductId	INT	Ідентифікатор продукції
4	DeliveryTerms	VARCHAR(1255)	Умови доставки
5	MinVolume	INT	Мінімальний обсяг постачання
6	MaxVolume	INT	Максимальний обсяг постачання
7	UnitPrice	DECIMAL(10,2)	Ціна за одиницю продукції

Таблиця «Suppliers» призначена для зберігання інформації про постачальників товарів та послуг. Вона є важливою для управління контактами постачальників, включаючи їхню назву, адресу, електронну пошту та телефон.

Таблиця 2.15 – Атрибути сутності «Suppliers»

№	Поле	Тип даних	Призначення поля
1	SupplierId	INT	Унікальний ідентифікатор постачальника
2	SupplierName	VARCHAR(255)	Назва постачальника
3	Address	VARCHAR(255)	Адреса постачальника
4	Email	VARCHAR(150)	Електронна пошта постачальника
5	Phone	VARCHAR(45)	Контактний телефон постачальника

Таблиця «Users» призначена для зберігання інформації про користувачів системи. Вона є важливою для управління доступом, реєстрації користувачів та надання відповідних ролей і прав доступу.

Таблиця 2.7 – Атрибути сутності «Users»

№	Поле	Тип даних	Призначення поля
1	UsersId	INT	Унікальний ідентифікатор користувача
2	FirstName	VARCHAR(45)	Ім'я користувача
3	LastName	VARCHAR(45)	Прізвище користувача
4	UsersName	VARCHAR(45)	Логін користувача
5	UsersPassword	VARCHAR(150)	Пароль користувача
6	RoleId	INT	Ідентифікатор ролі користувача
7	Description	TEXT	Опис користувача

Після визначення логічної структури бази даних здійснено перехід до побудови фізичної моделі бази даних, яка включає створення відповідних таблиць і встановлення зв'язків між ними. Фізична модель бази даних, яка представлена на рис. 2.8 відображає реальну реалізацію схеми даних, забезпечуючи зберігання, доступ і цілісність даних. Вона включає детальні специфікації всіх таблиць, їх атрибутів, типів даних, обмежень та зв'язків, що були описані в попередніх підрозділах.

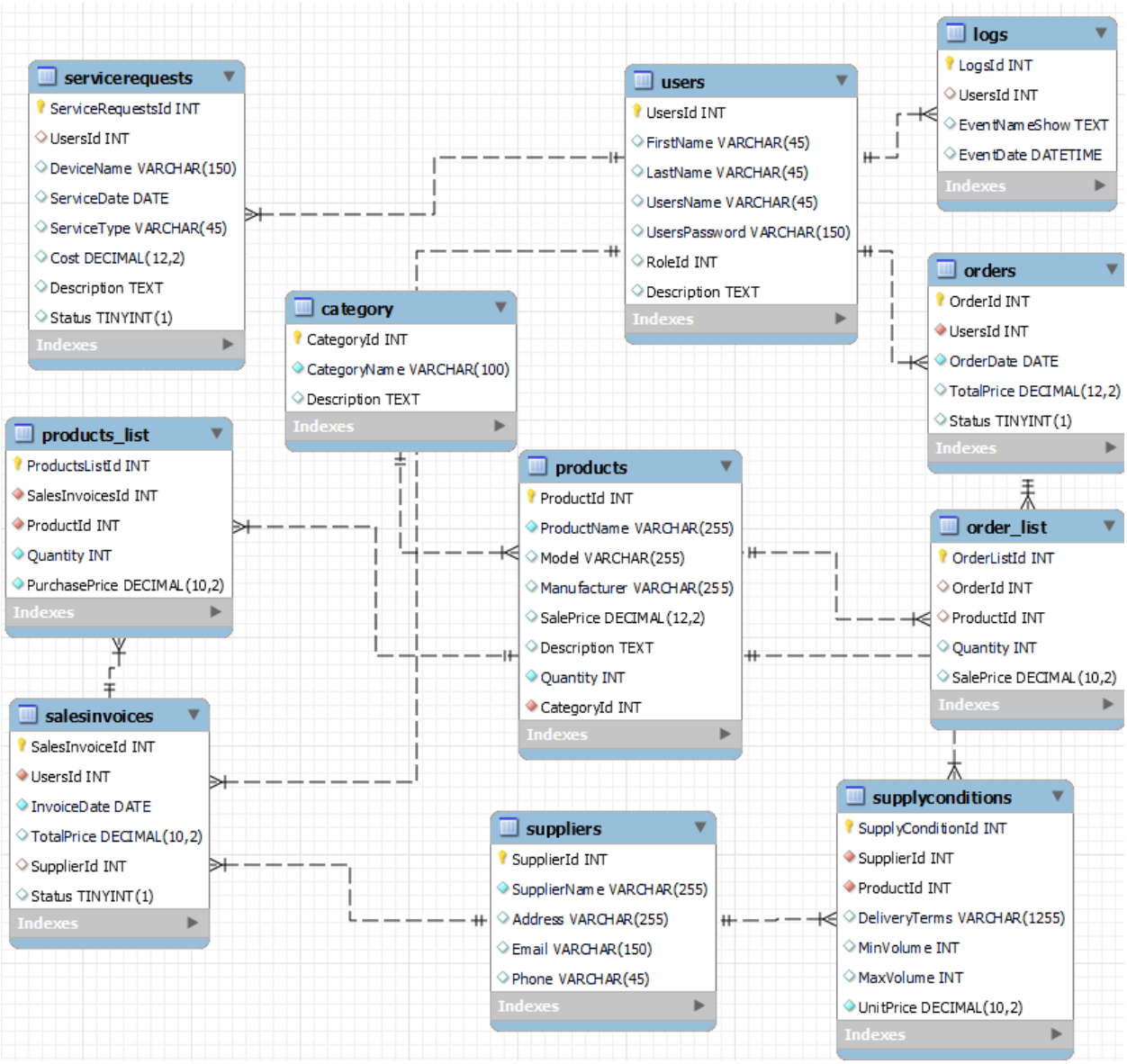


Рисунок 2.8 – Фізична модель бази даних

Після побудови фізичної моделі бази даних отримано повне уявлення про структуру та організацію даних у системі. Завершена модель бази даних стала основою для подальшої розробки програмного забезпечення та інтеграції всіх функціональних модулів системи «ТехноСервіс».

2.5 Архітектура проекту

Архітектура проекту є ключовим компонентом розробки інформаційної системи, що визначає структуру та взаємодію між різними її частинами. Для системи «ТехноСервіс» було обрано трирівневу архітектуру, яка включає рівні презентації, логіки бізнесу та даних. Така архітектура забезпечує чітке

розмежування функцій, що підвищує гнучкість, масштабованість та зручність підтримки системи. Вибір трирівневої архітектури обумовлений необхідністю ефективного управління складними бізнес-процесами, забезпечення безпеки даних та можливістю легкого оновлення окремих компонентів без порушення роботи всієї системи.

Було розроблено рівень даних системи, що зображений на рис. 2.9. Цей рівень відповідає за зберігання та управління даними, включаючи структури бази даних, таблиці та їх зв'язки. Рівень даних забезпечує цілісність, безпеку та ефективне оброблення даних, що є основою для функціонування всіх інших компонентів системи. Він включає всі необхідні механізми для зберігання, оновлення та отримання даних, що забезпечує стабільну роботу інформаційної системи «ТехноСервіс».

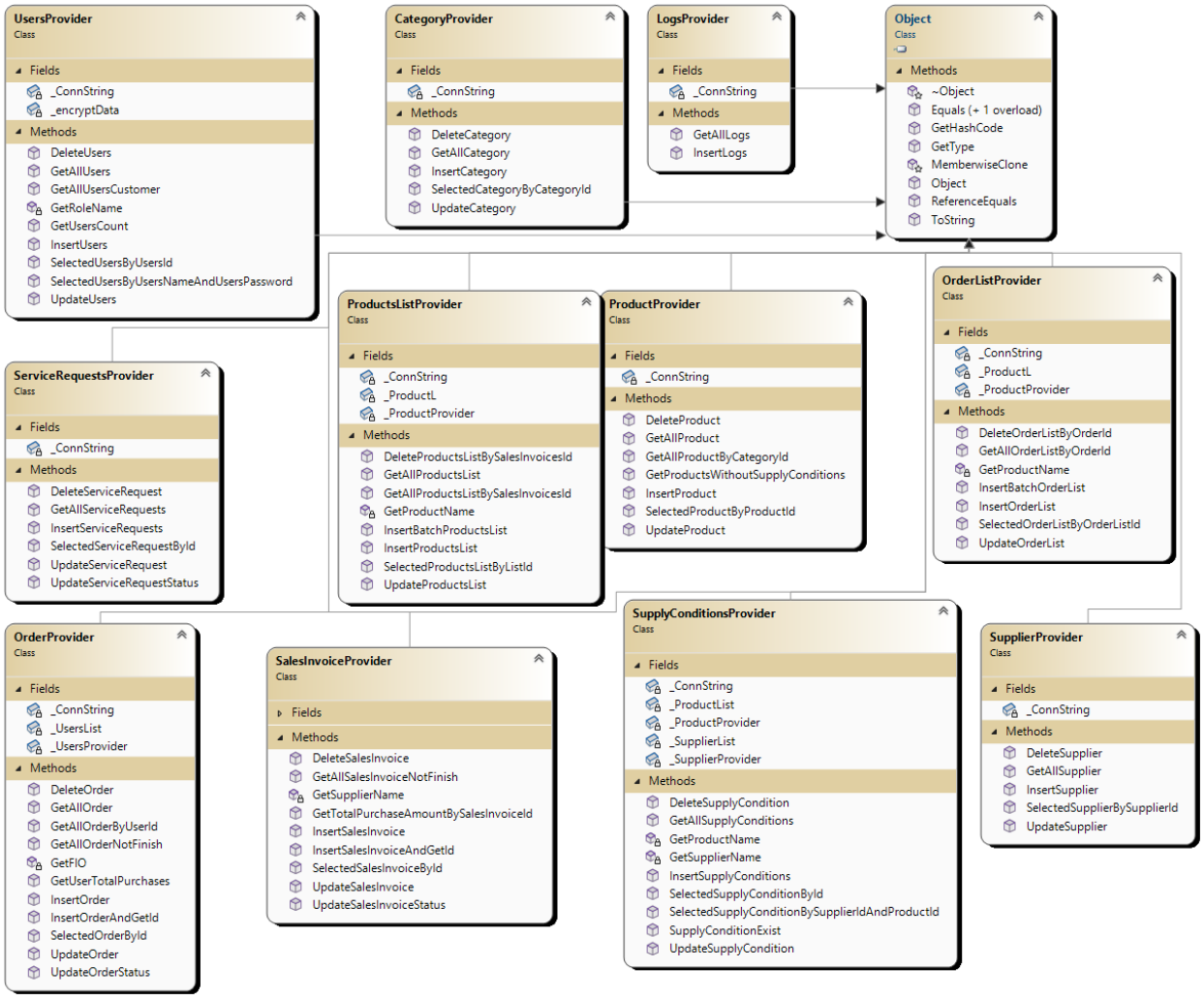


Рисунок 2.9 – Діаграма класів рівня даних системи

Діаграма класів цього рівня складається із:

– клас CategoryProvider відповідає за управління категоріями продукції. Він включає методи для створення нових категорій, оновлення даних існуючих категорій, видалення категорій з системи, а також методи для отримання детальної інформації про категорії, включно з фільтрацією за різними критеріями;

– клас LogsProvider призначений для управління подіями системи. Він забезпечує створення нових записів про події, оновлення даних існуючих записів, видалення подій з журналу та отримання інформації про події з можливістю фільтрації та сортування;

– клас OrderListProvider відповідає за управління списками куплених комплектуючих комп'ютерної техніки. Він включає методи для додавання нових позицій до списку, оновлення інформації про наявні позиції, видалення позицій зі списку та отримання детальної інформації про кожну позицію у списку;

– клас OrderProvider займається управлінням накладними на купівлю продукції. Він забезпечує створення нових накладних, оновлення даних існуючих накладних, видалення накладних з системи та отримання інформації про накладні з можливістю фільтрації за датою, статусом та іншими параметрами;

– клас ProductProvider відповідає за управління комплектуючими. Він включає методи для додавання нових комплектуючих до системи, оновлення даних про наявні комплектуючі, видалення комплектуючих з системи та отримання інформації про них з можливістю пошуку та фільтрації за різними критеріями;

– клас ProductsListProvider призначений для управління списками комплектуючих у постачальників. Він забезпечує додавання нових позицій до списків, оновлення інформації про наявні позиції, видалення позицій зі списків та отримання детальної інформації про кожну позицію у списку;

– клас SalesInvoiceProvider займається управлінням накладними на постачання. Він включає методи для створення нових накладних, оновлення

даних існуючих накладних, видалення накладних з системи та отримання інформації про накладні з можливістю фільтрації за датою, постачальником та іншими параметрами;

– клас ServiceRequestsProvider відповідає за управління запитами на ремонт техніки. Він забезпечує створення нових запитів, оновлення даних про наявні запити, видалення запитів з системи та отримання інформації про запити з можливістю фільтрації за датою, статусом та іншими параметрами;

– клас SupplierProvider призначений для управління постачальниками. Він включає методи для додавання нових постачальників до системи, оновлення даних про наявних постачальників, видалення постачальників з системи та отримання інформації про них з можливістю фільтрації за назвою, адресою та іншими критеріями;

– клас SupplyConditionsProvider займається управлінням умовами постачання. Він забезпечує створення нових умов постачання, оновлення даних про наявні умови, видалення умов з системи та отримання інформації про умови постачання з можливістю фільтрації за постачальником, продуктом та іншими параметрами;

– клас UsersProvider відповідає за управління користувачами системи. Він включає методи для створення нових користувачів, оновлення даних про наявних користувачів, видалення користувачів з системи та отримання інформації про користувачів з можливістю фільтрації за ролями, іменем та іншими критеріями.

Рівень користувацького інтерфейсу розробленої системи «ТехноСервіс» забезпечує зручний доступ до всіх функцій та даних системи через інтуїтивно зрозумілі форми та вікна. Він включає в себе різноманітні форми для взаємодії з користувачами, такі як перегляд замовлень, підтвердження купівлі, замовлення товарів, облік надходжень та управління ремонтами техніки. Розробка цього рівня спрямована на максимальну зручність користувачів, забезпечуючи швидкий доступ до необхідної інформації та виконання

операцій. На рис. 2.10 представлена діаграма класів цього рівня, яка демонструє основні компоненти інтерфейсу користувача.

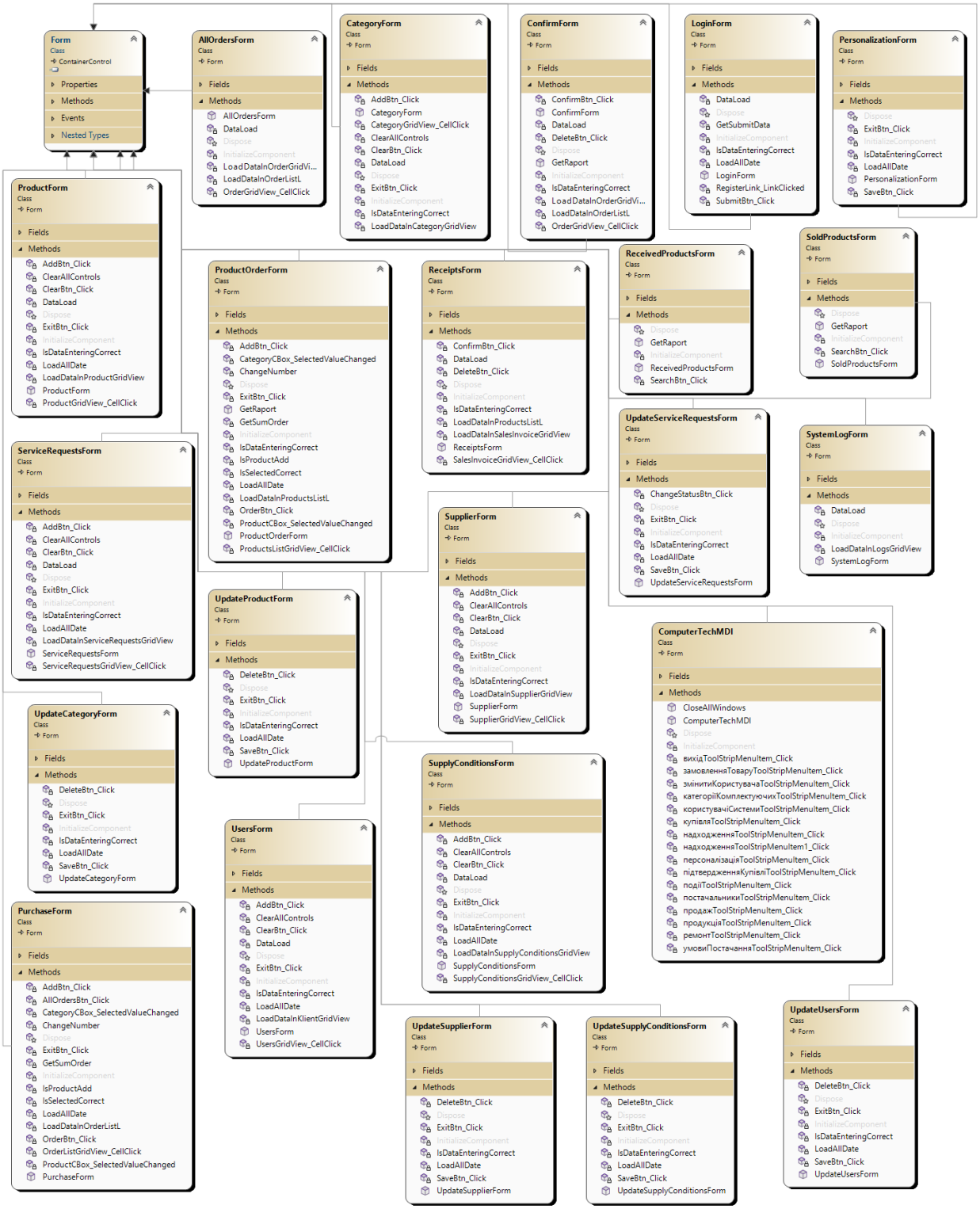


Рисунок 2.10 – Діаграма класів рівня користувацького інтерфейсу

Діаграма класів цього рівня складається із:

– ComputerTechMDI – головне вікно програми, яке забезпечує доступ до всіх основних функцій системи. Включає меню та панелі інструментів для навігації між різними формами та розділами системи;

– AllOrdersForm – форма для перегляду всіх замовлень користувача. Включає можливості фільтрації, сортування та детального перегляду інформації про кожне замовлення;

– ConfirmForm – форма для підтвердження купівлі продукції. Дозволяє користувачу перевірити деталі замовлення та підтвердити його виконання;

– ProductOrderForm – форма для створення нових замовлень на товари. Включає поля для введення інформації про товари, їх кількість та інші деталі замовлення;

– PurchaseForm – форма для купівлі продукції та формування накладної. Дозволяє вводити інформацію про постачальників, товари та кількість, а також генерувати накладні для купівлі;

– ReceiptsForm – форма для обліку надходжень продукції на підприємство. Включає можливості введення даних про постачання та їх підтвердження;

– ServiceRequestsForm – форма для додавання нових запитів на ремонт техніки та перегляду існуючих запитів. Дозволяє вводити деталі про техніку, тип ремонту та інші параметри;

– UpdateServiceRequestsForm – форма для редагування та видалення існуючих запитів на ремонт. Включає можливості оновлення даних про запит та видалення його з системи;

– CategoryForm – форма для додавання нових категорій комплектуючих та перегляду існуючих категорій. Включає поля для введення назви категорії та опису;

– ProductForm – форма для додавання нових комплектуючих та перегляду існуючої продукції. Дозволяє вводити деталі про продукцію, такі як назва, модель, виробник, ціна та кількість;

– LoginForm – форма для авторизації користувачів в системі. Дозволяє вводити облікові дані (ім'я користувача та пароль) та забезпечує доступ до системи після успішної аутентифікації;

– PersonalizationForm – форма для налаштування персональних параметрів користувача. Дозволяє змінювати особисті налаштування інтерфейсу, такі як тема, мова та інші параметри;

– SystemLogForm – форма для перегляду подій системи. Дозволяє моніторити активність користувачів, аналізувати безпекові події та відстежувати зміни, що відбуваються в системі;

– UpdateUsersForm – форма для редагування та видалення даних про користувачів. Включає можливості оновлення інформації про користувача та видалення користувачів з системи;

– UsersForm – форма для додавання нових користувачів та перегляду існуючих користувачів системи. Дозволяє вводити дані про користувачів, такі як ім'я, прізвище, роль та облікові дані;

– SupplierForm – форма для додавання нових постачальників та перегляду інформації про існуючих постачальників. Включає поля для введення назви постачальника, адреси, контактних даних та інших важливих деталей;

– SupplyConditionsForm – форма для додавання нових умов постачання та перегляду існуючих умов. Дозволяє вводити дані про умови доставки, мінімальні та максимальні обсяги, а також ціну за одиницю продукції;

– UpdateCategoryForm – форма для редагування та видалення категорій комплектуючих. Включає можливості оновлення назви категорії, опису та видалення категорії з системи;

– UpdateProductForm – форма для редагування та видалення інформації про продукцію. Дозволяє змінювати дані про продукцію, такі як назва, модель, виробник, ціна та кількість, а також видаляти продукцію з системи;

– UpdateSupplierForm – форма для редагування та видалення інформації про постачальників. Включає можливості оновлення контактних даних,

адреси та інших важливих деталей постачальника, а також видалення постачальника з системи;

– UpdateSupplyConditionsForm – форма для редагування та видалення умов постачання. Дозволяє змінювати дані про умови доставки, обсяги постачання та ціну за одиницю продукції, а також видаляти умови постачання з системи;

– ReceivedProductsForm – форма для генерації звітів про надходження продукції на підприємство. Дозволяє переглядати інформацію про всі отримані товари, включаючи дати, постачальників та кількість продукції;

– SoldProductsForm – форма для генерації звітів про продану продукцію. Дозволяє переглядати інформацію про всі продані товари, включаючи дати продажу, кількість, ціни та покупців.

Бізнес-логіка є ключовим компонентом інформаційної системи «ТехноСервіс», який відповідає за реалізацію основних функціональних можливостей системи та забезпечує взаємодію між рівнем даних і користувацьким інтерфейсом. Вона включає в себе всі правила та алгоритми, що регулюють бізнес-процеси, такі як валідація даних, шифрування інформації, управління ролями користувачів та генерація звітів. Бізнес-логіка забезпечує цілісність і узгодженість даних, а також їх правильну обробку відповідно до визначених правил і процедур. Завдяки чітко визначеній бізнес-логіці, система може ефективно підтримувати різні операції та забезпечувати високу якість обслуговування клієнтів.

На рис. 2.11 представлена структура бізнес-логіки, яка демонструє взаємодію між різними класами, що реалізують основні функції системи.

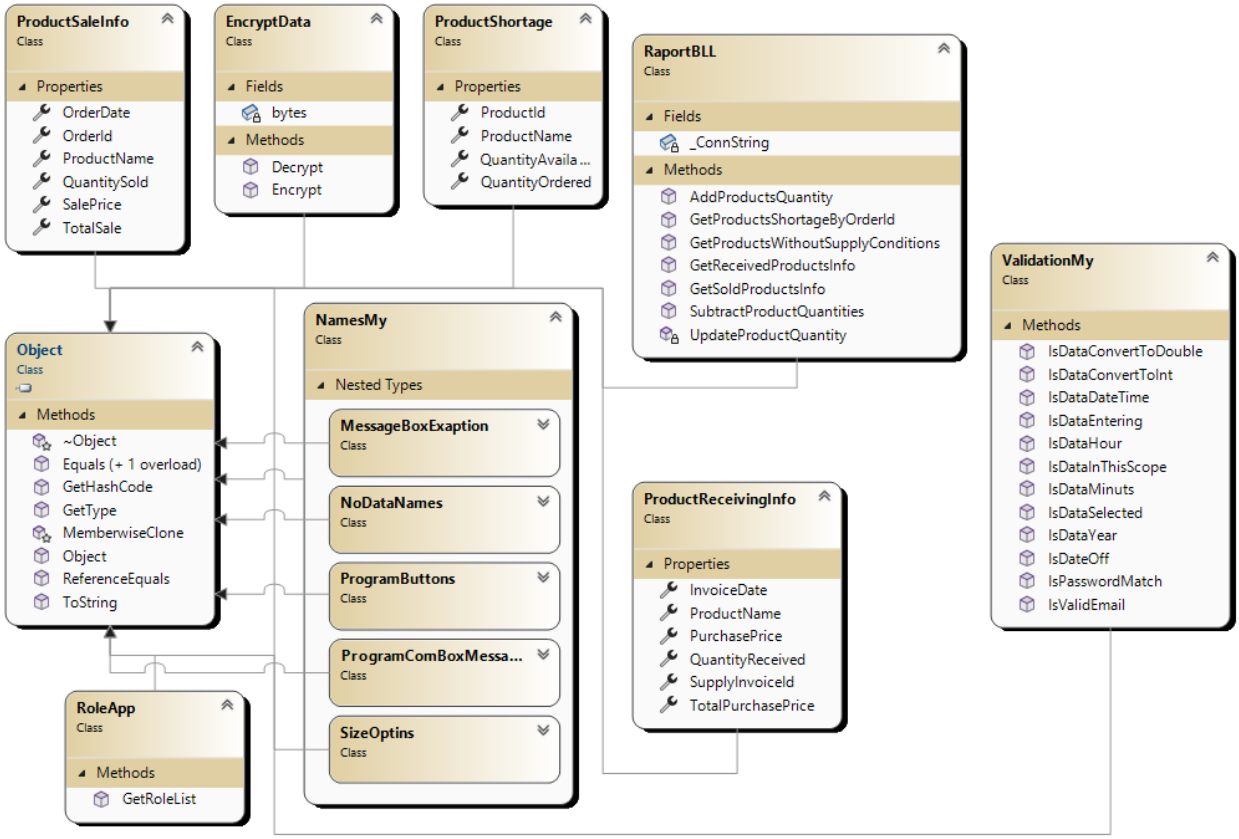


Рисунок 2.11 – Діаграма класів бізнес-логіки

Діаграма класів цього рівня складається із:

– клас ValidationMy відповідає за перевірку коректності введених даних у системі. Він включає методи для валідації різних типів даних, таких як текстові поля, числові значення, дати, електронні адреси та інші. Цей клас забезпечує, щоб всі дані, які вводяться користувачами, відповідали заданим критеріям і стандартам, зменшуючи ймовірність помилок та некоректної інформації;

– клас NamesMy використовується для обробки та нормалізації імен у системі. Він включає методи для форматування, порівняння та стандартного представлення імен користувачів та інших сутностей. Цей клас забезпечує єдиний підхід до роботи з іменами, що допомагає уникнути дублювання та помилок при введенні імен;

– клас EncryptData відповідає за шифрування та дешифрування даних у системі. Він включає методи для захисту конфіденційної інформації, такої як паролі користувачів, особисті дані та інші критичні дані. Використання цього

класу допомагає забезпечити безпеку даних, зберігаючи їх у захищеному вигляді як під час зберігання, так і при передачі;

– клас RoleApp призначений для управління ролями та правами доступу користувачів у системі. Він включає методи для створення, редагування та видалення ролей, а також для призначення прав доступу до різних функцій системи. Цей клас забезпечує гнучке управління доступом, дозволяючи налаштовувати рівні доступу для різних категорій користувачів відповідно до їхніх обов'язків та прав;

– клас RaportBLL відповідає за генерацію та управління звітами у системі. Він включає методи для створення різних типів звітів, таких як фінансові, операційні та аналітичні звіти. Клас RaportBLL забезпечує доступ до необхідних даних, їх обробку та представлення у зручному для користувачів вигляді;

– клас ProductShortage відповідає за управління інформацією про нестачу продукції. Він включає методи для аналізу поточних запасів та виявлення продукції, яка потребує поповнення. Клас дозволяє генерувати повідомлення та звіти про нестачу продукції, що допомагає вчасно реагувати на критичні ситуації та забезпечувати безперебійне постачання товарів;

– клас ProductSaleInfo призначений для управління інформацією про продаж продукції. Він включає методи для збору та обробки даних про продажі, аналізу продажних тенденцій та генерації відповідних звітів. Клас допомагає відстежувати ефективність продажів, виявляти найбільш популярні продукти та приймати обґрунтовані рішення щодо стратегії продажу;

– клас ProductReceivingInfo відповідає за управління інформацією про надходження продукції. Він включає методи для реєстрації нових надходжень, перевірки відповідності поставок замовленням та оновлення даних про запаси. Клас забезпечує точний облік надходжень, що допомагає підтримувати актуальну інформацію про запаси та покращує процеси управління постачаннями.

2.6 Висновок

У рамках даного розділу було виконано комплексний аналіз та вибір технологій для реалізації інформаційної системи «ТехноСервіс». Проведено порівняння мов програмування Python, Java та C#, в результаті якого обрано мову C# завдяки її високій продуктивності та можливостям інтеграції. Також проаналізовано СУБД MS SQL Server, MySQL та Oracle, в результаті чого обрано MySQL через її високу гнучкість та низьку вартість володіння.

Визначено функціональні та нефункціональні вимоги до системи, а також побудовано діаграми прецедентів для основних ролей користувачів: директор, робітник, продавець та покупець. Це дозволило чітко окреслити цілі та завдання кожної ролі у системі, а також описати варіанти використання системи.

Проведено моделювання основних бізнес-процесів, включаючи діаграми послідовностей для процесу додавання умов постачання продукції, формування накладної для купівлі комплектуючих та формування звітності. Це дозволило виявити ключові етапи роботи та оптимізувати їх для підвищення загальної ефективності підприємства.

Розроблено та описано структуру бази даних, включаючи детальний опис таблиць та побудову фізичної моделі бази даних. Це забезпечує цілісність і узгодженість даних, а також їх правильну обробку відповідно до визначених правил і процедур.

Обрано трирівневу архітектуру проекту, що включає рівні презентації, логіки бізнесу та даних. Побудовано та описано класи кожного рівня, що забезпечує чітке розмежування функцій, підвищує гнучкість, масштабованість та зручність підтримки системи.

Результати, отримані в цьому розділі, дозволять у наступному розділі перейти до реалізації системи, забезпечуючи відповідність усім визначеним вимогам та ефективну інтеграцію усіх компонентів.

3 РОЗРОБКА, ТЕСТУВАННЯ ТА ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМИ

3.1 Розробка модулів програмного забезпечення

Розробка модулів програмного забезпечення є критичним етапом у створенні інформаційної системи, що автоматизує взаємодію з клієнтами в продуктовому магазині. На цьому етапі здійснюється аналіз вимог до системи, проектування архітектури та програмування основних компонентів, які забезпечуватимуть належну функціональність та інтеграцію з існуючими бізнес-процесами. Особливу увагу приділяють налаштуванню параметрів підключення до бази даних, оскільки це впливає на швидкість і надійність роботи всієї системи. Одним з ключових аспектів є правильне налаштування параметрів підключення до бази даних MySQL, що дозволяє забезпечити ефективну роботу з даними та уникнути проблем з доступом. На рис. 3.1 зображено основні параметри підключення до бази даних MySQL, які використовуються в розроблюваній системі.

```
<appSettings>
  <!-- Підключення до бази даних MySQL -->
  <add key="CONNECTSQL"
       value="server=localhost;user=root;database=shop;password=12345;" />
</appSettings>
```

Рисунок 3.1 – Параметри підключення до бази даних

У фрагменті коду в розділі <appSettings> знаходиться параметр підключення до бази даних MySQL. Цей параметр визначається ключем «CONNECTSQL» та містить значення, яке складається з кількох складових. Сервер підключення вказано як «localhost», що означає використання локального сервера для доступу до бази даних. Ім'я користувача встановлено як «root», що є стандартним ім'ям користувача для адміністрування бази даних MySQL. Назва бази даних, до якої здійснюється підключення, визначена як «shop», а пароль для доступу до бази даних встановлений на «12345».

На початку розробки системи було створено Головне вікно, яке забезпечує доступ до основних функцій через зручне меню. Це вікно слугує

центральною точкою навігації, де користувач може швидко обрати необхідний розділ чи функцію системи. На рис. 3.2 наведено зображення головного вікна системи з меню, яке ілюструє його структуру та основні елементи.

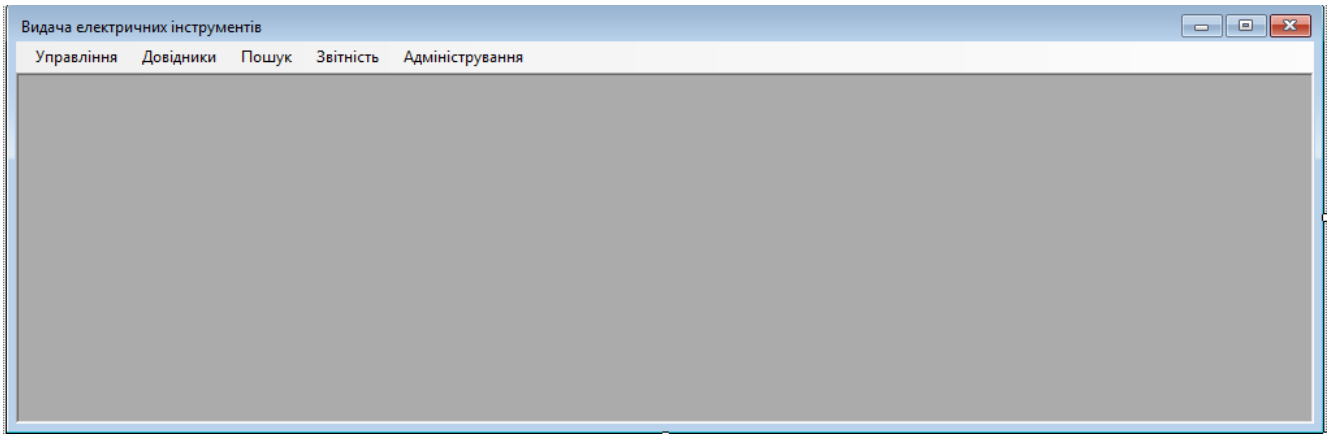


Рисунок 3.2 – Головне вікно системи

На кожен пункт меню доданий код для перевірки ролі користувача, що забезпечує доступ до відповідних функцій лише авторизованим користувачам з необхідними правами. Після перевірки ролі користувача здійснюється виклик відповідної форми, що відповідає обраному пункту меню. Приклад цього коду наведено на рис. 3.3, де продемонстровано логіку перевірки та виклику форми.

```
private void купівляToolStripMenuItem_Click(object sender, EventArgs e)
    if (LoginForm.CurrentUser.RoleId == 4) {
        CloseAllWindows();
        PurchaseForm purchaseForm = new PurchaseForm();
        purchaseForm.MdiParent = this;
        purchaseForm.WindowState = FormWindowState.Maximized;
        purchaseForm.Show();
    } else {
        MessageBox.Show(NamesMy.MessageBoxExaption.YouDontHavePermission);
    }
}
```

Рисунок 3.3 – Приклад виклику меню

У методі на рис. 3.3 здійснюється обробка події кліку на пункт меню «Купівля». Спочатку перевіряється роль поточного користувача за допомогою умови if (LoginForm.CurrentUser.RoleId == 4). Якщо роль користувача відповідає 4, то закриваються всі відкриті вікна методом CloseAllWindows. Потім створюється новий екземпляр форми PurchaseForm, встановлюється її батьківське вікно (MdiParent) на поточне вікно, задається стан вікна

(WindowState) на максимізований, і форма відображається за допомогою методу Show. Якщо роль користувача не відповідає 4, то відображається повідомлення з помилкою, використовуючи MessageBox.Show та вказуючи текст повідомлення з YouDontHavePermission.

При розробці форми «Продукція» було створено інтерфейс для введення та редагування даних про продукцію в системі. Форма включає поля для вибору категорії, введення назви, моделі, виробника, ціни продажу, кількості та опису продукції, що забезпечує повноту інформації для кожного товару. Кнопки «Додати», «Очистити» та «Вихід» дозволяють користувачам додавати нові записи, очищати введені дані та виходити з форми відповідно. На рис. 3.4 наведено зображення форми «Продукція», яке ілюструє її основні елементи та функціональні можливості.

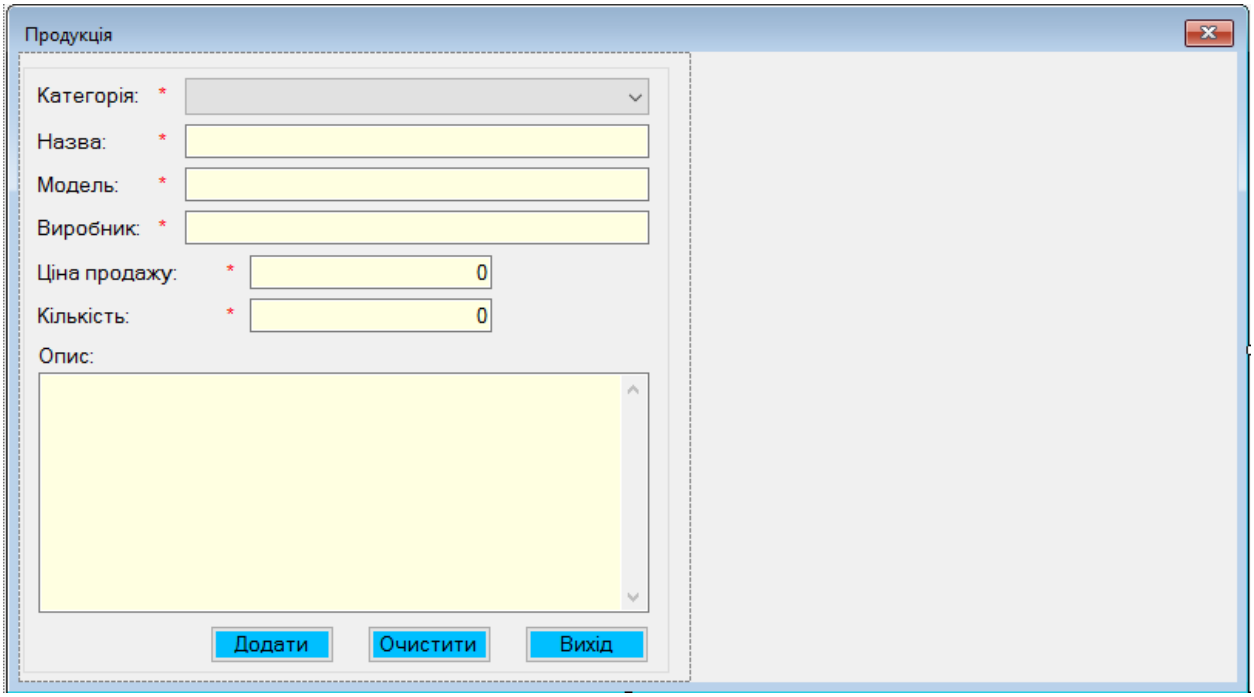


Рисунок 3.4 – Реалізація форми опрацювання даних продукції

У конструкторі форми викликається метод «LoadAllDate», який призначений для завантаження даних категорій продукції (рис. 3.5).

```
1 reference
private void LoadAllDate() {
    _CategoryList = _CategoryProvider.GetAllCategory();
    CategoryCBox.DataSource = _CategoryList;
    CategoryCBox.ValueMember = "CategoryId";
    CategoryCBox.DisplayMember = "CategoryName";
}
```

Рисунок 3.5 – Завантаження даних про категорії продукції

У методі LoadAllDate на початку викликається метод «GetAllCategory», результатом якого є список категорій, що присвоюється змінній _CategoryList. Далі цей список встановлюється як джерело даних для комбобоксу CategoryCBox. Встановлюються властивості ValueMember і DisplayMember для комбобоксу, що дозволяють відображати назви категорій і використовувати їхні ідентифікатори.

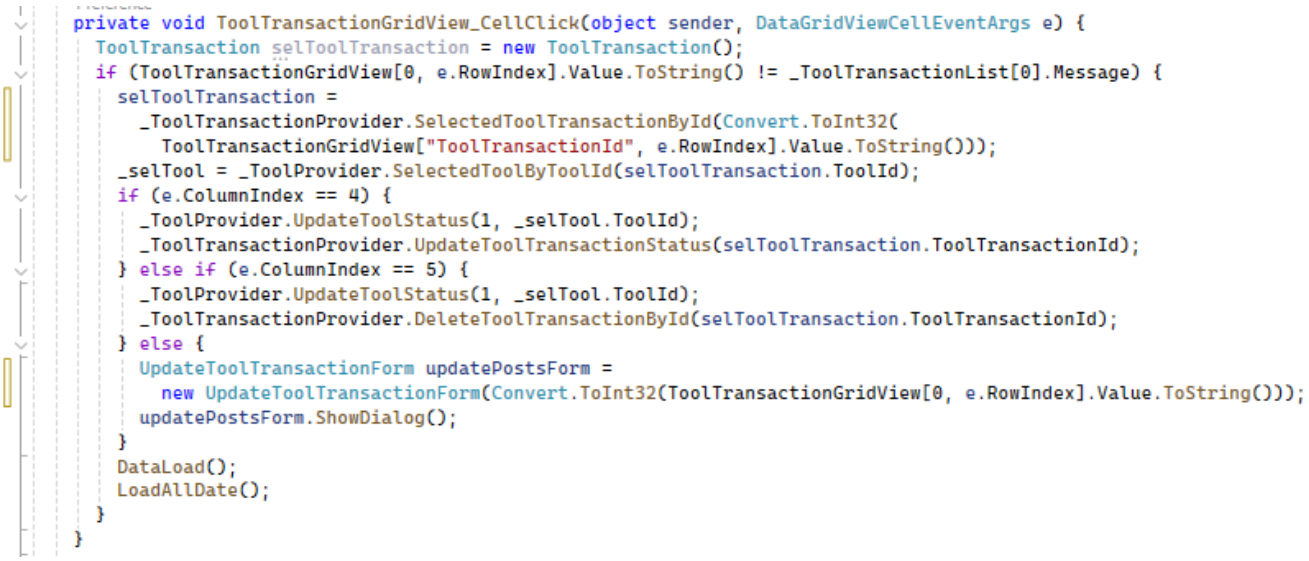
На рис. 3.6 наведено код методу «AddBtn_Click» у якому здійснюється обробка події натискання на кнопку «Додати».

```
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    _ProductPrivider.InsertProduct(ProductNameTBox.Text, ModelTBox.Text, ManufacturerTBox.Text,
      Convert.ToDouble(SalePriceTBox.Text),  DescriptionTBox.Text,
      Convert.ToInt32(QuantityTBox.Text), Convert.ToInt32(CategoryCBox.SelectedValue));
    DataLoad();
    ClearAllControls();
  }
}
```

Рисунок 3.6 – Код методу додавання нової комплектуючої

Спочатку у методі перевіряється правильність введених даних за допомогою методу «IsDataEnteringCorrect». Якщо дані введені коректно, викликається метод «InsertProduct» для додавання нового продукту в базу даних. Параметри для цього методу включають значення текстових полів ProductNameTBox, ModelTBox, ManufacturerTBox, SalePriceTBox, DescriptionTBox, QuantityTBox та вибраного значення з комбобоксу CategoryCBox. Після успішного додавання даних викликається метод DataLoad для оновлення даних у системі та метод «ClearAllControls» для очищення всіх контролів на формі.

Також розроблено метод, який обробляє події кліку на комірки у таблиці де відображається вся комп’ютерна техніка (рис. 3.7).



```
private void ToolTransactionGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
  ToolTransaction selToolTransaction = new ToolTransaction();
  if (ToolTransactionGridView[0, e.RowIndex].Value.ToString() != _ToolTransactionList[0].Message) {
    selToolTransaction =
      _ToolTransactionProvider.SelectedToolTransactionById(Convert.ToInt32(
        ToolTransactionGridView["ToolTransactionId", e.RowIndex].Value.ToString()));
    _selTool = _ToolProvider.SelectedToolByToolId(selToolTransaction.ToolId);
    if (e.ColumnIndex == 4) {
      _ToolProvider.UpdateToolStatus(1, _selTool.ToolId);
      _ToolTransactionProvider.UpdateToolTransactionStatus(selToolTransaction.ToolTransactionId);
    } else if (e.ColumnIndex == 5) {
      _ToolProvider.UpdateToolStatus(1, _selTool.ToolId);
      _ToolTransactionProvider.DeleteToolTransactionById(selToolTransaction.ToolTransactionId);
    } else {
      UpdateToolTransactionForm updatePostsForm =
        new UpdateToolTransactionForm(Convert.ToInt32(ToolTransactionGridView[0, e.RowIndex].Value.ToString()));
      updatePostsForm.ShowDialog();
    }
    DataLoad();
    LoadAllDate();
  }
}
```

Рисунок 3.7 – Код події для управління виданими інструментами

У методі «ProductGridView_CellClick» обробляється подія кліку по комірці в таблиці ProductGridView. Спочатку перевіряється, чи індекс рядка, по якому було здійснено клік, є дійсним та чи значення першої комірки цього рядка не дорівнює повідомленню зі списку продуктів. Якщо умови виконуються, зберігається індекс вибраного рядка в змінній _selectedRowIndex. Потім створюється новий екземпляр форми «UpdateProductForm», в який передається ідентифікатор продукту з вибраного рядка таблиці. Форма «UpdateProductForm» відкривається у вигляді діалогового вікна. Після закриття форми викликається метод DataLoad для оновлення даних у таблиці.

Для формування складної логіки у системі розроблено ряд методів із комплексними запитами до бази даних. Зокрема на рис. 3.8 показано запит для формування статистики по дефіцитній комп’ютерній техніці.

```
string query = @"
        SELECT p.ProductId, p.ProductName, ol.Quantity AS QuantityOrdered, p.Quantity AS QuantityAvailable
        FROM order_list ol
        JOIN products p ON ol.ProductId = p.ProductId
        WHERE ol.OrderId = @OrderId AND ol.Quantity > p.Quantity;
    ";
```

Рисунок 3.8 – Код запиту формування статистики по дефіцитній комп’ютерній техніці

Метод «GetProductsShortageByOrderId» приймає параметр orderId та повертає список об'єктів типу ProductShortage, що представляють дефіцитні

продукти для заданого замовлення. Всередині методу створюється новий список shortages для зберігання результатів. За допомогою об'єкта MySqlConnection, який ініціалізується рядком підключення _ConnString, здійснюється підключення до бази даних. SQL-запит, визначений у рядку query, вибирає ідентифікатор продукту, назву продукту, кількість замовлених одиниць та кількість доступних одиниць з таблиці order_list і products. Запит включає умову, яка перевіряє, чи кількість замовлених одиниць перевищує доступну кількість для кожного продукту в заданому замовленні.

Метод «GetSoldProductsInfo» приймає два параметри, startDate і endDate, та повертає список об'єктів типу ProductSaleInfo, що містять інформацію про продажі продуктів у заданому періоді часу (рис. 3.9).

```
string query = @"
        SELECT o.OrderId, o.OrderDate, p.ProductName, ol.Quantity, ol.SalePrice,
        (ol.Quantity * ol.SalePrice) AS TotalSale
        FROM orders o
        JOIN order_list ol ON o.OrderId = ol.OrderId
        JOIN products p ON ol.ProductId = p.ProductId
        WHERE o.Status = 1 AND o.OrderDate BETWEEN @StartDate AND @EndDate
        ORDER BY o.OrderDate ASC";
```

Рисунок 3.9 – Фрагмент коду «GetSoldProductsInfo»

Всередині методу створюється новий список salesInfo для зберігання результатів. Використовуючи об'єкт MySqlConnection, який ініціалізується рядком підключення _ConnString, здійснюється підключення до бази даних. SQL-запит, визначений у рядку query, вибирає ідентифікатори замовлень, дати замовлень, назви продуктів, кількість проданих одиниць, ціну продажу за одиницю та загальну суму продажу для кожного продукту з таблиць orders, order_list та products. Запит включає умову, яка перевіряє, чи статус замовлення дорівнює 1 (що вказує на завершене замовлення) і чи дата замовлення знаходиться у межах заданого періоду. Дані відсортовуються за датою замовлення у порядку зростання. Параметри запиту (@StartDate і @EndDate) заповнюються значеннями, переданими в метод.

Метод «GetReceivedProductsInfo» приймає два параметри, startDate і endDate, та повертає список об'єктів типу ProductReceivingInfo, що містять інформацію про отримані продукти у заданому періоді (рис. 3.10).

```
string query = @"
        SELECT si.SalesInvoiceId, si.InvoiceDate, p.ProductName, pl.Quantity AS QuantityReceived, pl.PurchasePrice
        FROM salesinvoices si
        JOIN products_list pl ON si.SalesInvoiceId = pl.SalesInvoicesId
        JOIN products p ON pl.ProductId = p.ProductId
        WHERE si.InvoiceDate BETWEEN @StartDate AND @EndDate
        ORDER BY si.InvoiceDate ASC";
```

Рисунок 3.10 – Фрагмент код методу «GetReceivedProductsInfo»

Всередині методу створюється новий список receivingInfo для зберігання результатів. Використовуючи об'єкт MySqlConnection, який ініціалізується рядком підключення _ConnString, здійснюється підключення до бази даних. SQL-запит, визначений у рядку query, вибирає ідентифікатори накладних, дати накладних, назви продуктів, кількість отриманих одиниць та ціну покупки за одиницю з таблиць salesinvoices, products_list та products. Запит включає умову, яка перевіряє, чи дата накладної знаходиться у межах заданого періоду. Дані відсортовуються за датою накладної у порядку зростання.

3.2 Функціональне та модульне тестування

Функціональне та модульне тестування є важливими етапами розробки програмного забезпечення, що забезпечують його якість та надійність. Функціональне тестування дозволяє перевірити відповідність роботи системи заданим вимогам, зосереджуючись на правильності виконання її основних функцій. Модульне тестування, у свою чергу, спрямоване на перевірку окремих компонентів системи, що дозволяє виявити та усунути помилки на ранніх етапах розробки. Особливу увагу приділяють створенню тестових сценаріїв, які охоплюють різні аспекти роботи інтерфейсу та бізнес-логіки. Одним з прикладів є тестування форми «Комп'ютерна техніка», для якої розроблено низку сценаріїв перевірки основних функцій.

У табл. 3.1 наведено детальний опис тестових сценаріїв для цієї форми, що включають перевірку коректності введення даних, взаємодію з базою даних та обробку різних подій.

Таблиця 3.1 – Тестові сценарії для форми «Комп'ютерна техніка»

№	Сценарій	Вхідні дані	Очікуваний результат
1	2	3	4
1	Введення всіх обов'язкових полів та натискання кнопки «Додати»	Категорія: Відеокарти; Назва: AMD Radeon RX; Модель: RX 570; Виробник: AMD; Ціна продажу: 7000; Кількість: 20	Новий продукт успішно додається до таблиці, відображається в списку продуктів.
2	Введення неповних даних та натискання кнопки «Додати»	Категорія: Відеокарти; Назва: AMD Radeon RX; Модель: (порожнє); Виробник: AMD; Ціна продажу: 7000; Кількість: 20	Відображається повідомлення про помилку, яке інформує користувача про необхідність заповнення всіх обов'язкових полів.
3	Введення некоректних даних та натискання кнопки «Додати»	Категорія: Відеокарти; Назва: AMD Radeon RX; Модель: RX 570; Виробник: AMD; Ціна продажу: -7000; Кількість: -20	Відображається повідомлення про помилку, яке інформує користувача про некоректні дані у полях «Ціна продажу» та «Кількість».
4	Очищення всіх полів за допомогою кнопки «Очистити»	Будь-які дані, введені у форму	Всі поля форми очищуються, жодні дані не зберігаються.
5	Натискання кнопки «Вихід»	Будь-які дані, введені у форму	Закриття форми, повернення до попереднього вікна або

Результат виконання тестових сценарії представлено на рис. 3.11.

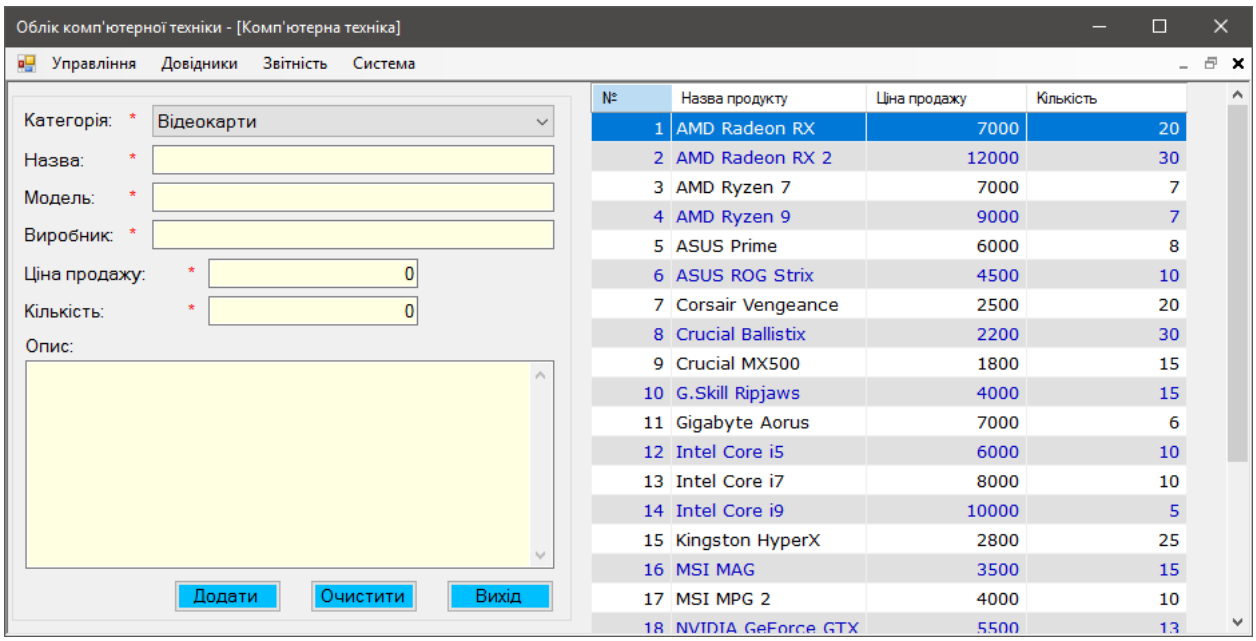


Рисунок 3.11 – Результати тестування форми «Комп'ютерна техніка»

Форма «Купівля» є важливим елементом системи, яка дозволяє покупцям здійснювати замовлення комп’ютерної техніки підприємства. У табл. 3.2 наведено тестові сценарії для цієї форми.

Таблиця 3.2 – Тестові сценарії для форми «Купівля»

№	Сценарій	Вхідні дані	Очікуваний результат
№	Сценарій	Вхідні дані	Очікуваний результат
1	Введення всіх обов'язкових полів та натискання кнопки «Додати»	Категорія: Оперативна пам'ять; Товар: Corsair Vengeance; Кількість: 2	Товар успішно додається до списку замовлень, відображається в таблиці праворуч.
2	Введення неповних даних та натискання кнопки «Додати»	Категорія: Оперативна пам'ять; Товар: (не вибрано); Кількість: 2	Відображається повідомлення про помилку, яке інформує користувача про необхідність заповнення всіх обов'язкових полів.
3	Введення некоректних даних та натискання кнопки «Додати»	Категорія: Оперативна пам'ять; Товар: Corsair Vengeance; Кількість: -2	Відображається повідомлення про помилку, яке інформує користувача про некоректні дані у полі «Кількість».

4	Натискання кнопки «Видалити» для обраного товару у таблиці	Будь-який товар, доданий до таблиці праворуч	Обраний товар видаляється зі списку замовлень, таблиця оновлюється.

Продовження табл. 3.2

5	Натискання кнопки «Замовити» при наявності товарів у списку замовлень	Будь-які товари, додані до списку замовлень	Замовлення обробляється, з'являється підтвердження успішного замовлення.
6	Натискання кнопки «Замовити» при відсутності товарів у списку замовлень	Порожній список замовлень	Відображається повідомлення про помилку, яке інформує користувача про необхідність додати товари до списку замовлень перед замовленням.

На рис. 3.12 представлено результати виконання тестових сценарії для форми «Купівля».

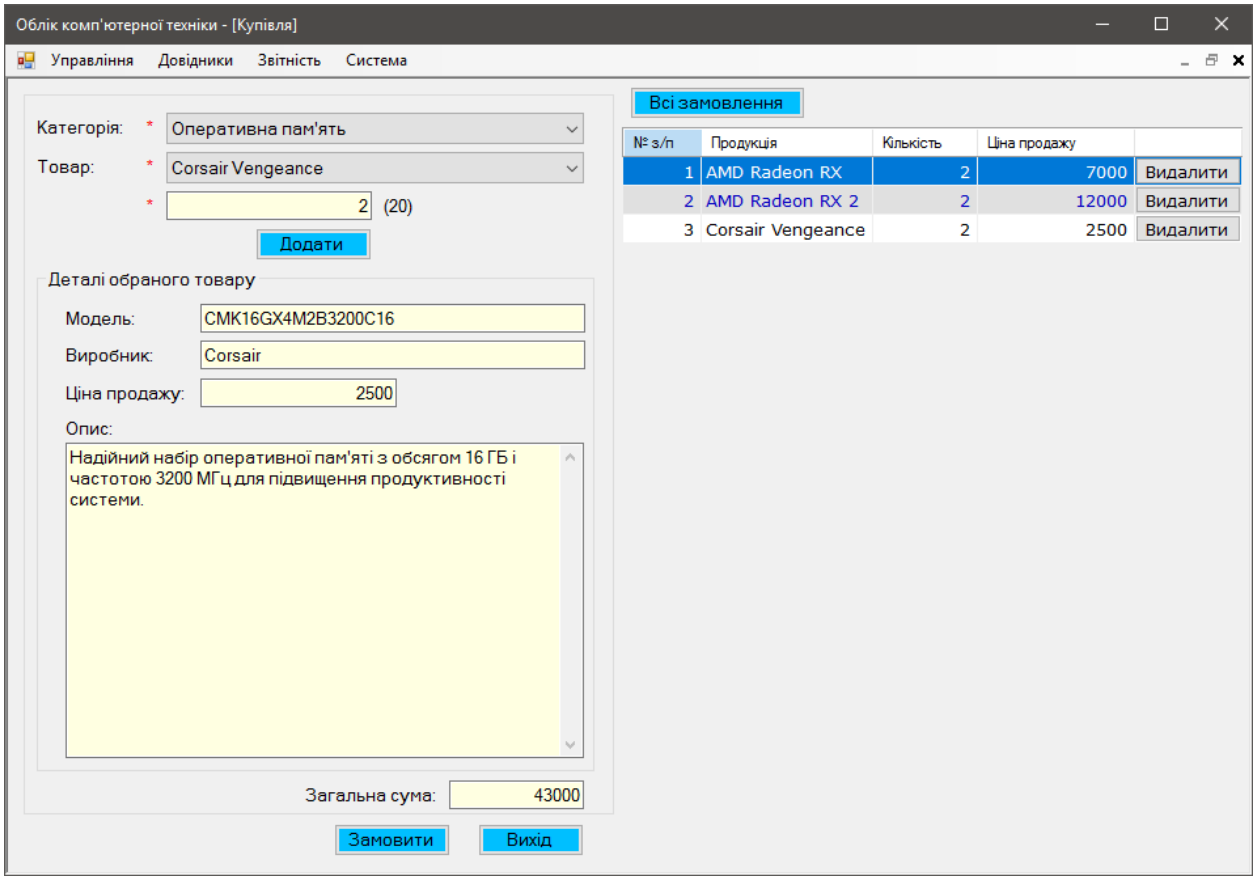


Рисунок 3.12 – Результати тестування форми «Купівля»

Форма «Звітність по проданій продукції» дозволяє користувачам формувати звіт про продаж продукції в заданий період часу. У табл. 3.3 представлено тестові сценарії для цієї форми.

Таблиця 3.3 – Тестові сценарії для форми «Звітність по проданій продукції»

№	Сценарій	Вхідні дані	Очікуваний результат
1	Введення коректного періоду та натискання кнопки «Формувати»	Початок періоду: 1 квітня 2024 р.; Кінець періоду: 10 червня 2024 р.	Генерується звіт з проданою продукцією за обраний період, відображається у таблиці нижче.
2	Введення некоректного періоду (кінець періоду раніше початку періоду) та натискання кнопки «Формувати»	Початок періоду: 10 червня 2024 р.; Кінець періоду: 1 квітня 2024 р.	Відображається повідомлення про помилку, яке інформує користувача про некоректний вибір періоду.
3	Введення частково заповнених полів (одне з полів не заповнене) та натискання кнопки «Формувати»	Початок періоду: 1 квітня 2024 р.; Кінець періоду: (порожнє)	Відображається повідомлення про помилку, яке інформує користувача про необхідність заповнення обох полів.
4	Введення періоду без проданої продукції та натискання кнопки «Формувати»	Початок періоду: 1 січня 2024 р.; Кінець періоду: 31 січня 2024 р.	Відображається порожній звіт або повідомлення про відсутність даних за обраний період.
5	Натискання кнопки «Вихід»	Будь-які дані, введені у форму	Закриття форми, повернення до попереднього вікна або головного меню.

На рис. 3.13 представлено результати виконання тестових сценарії для цієї форми.

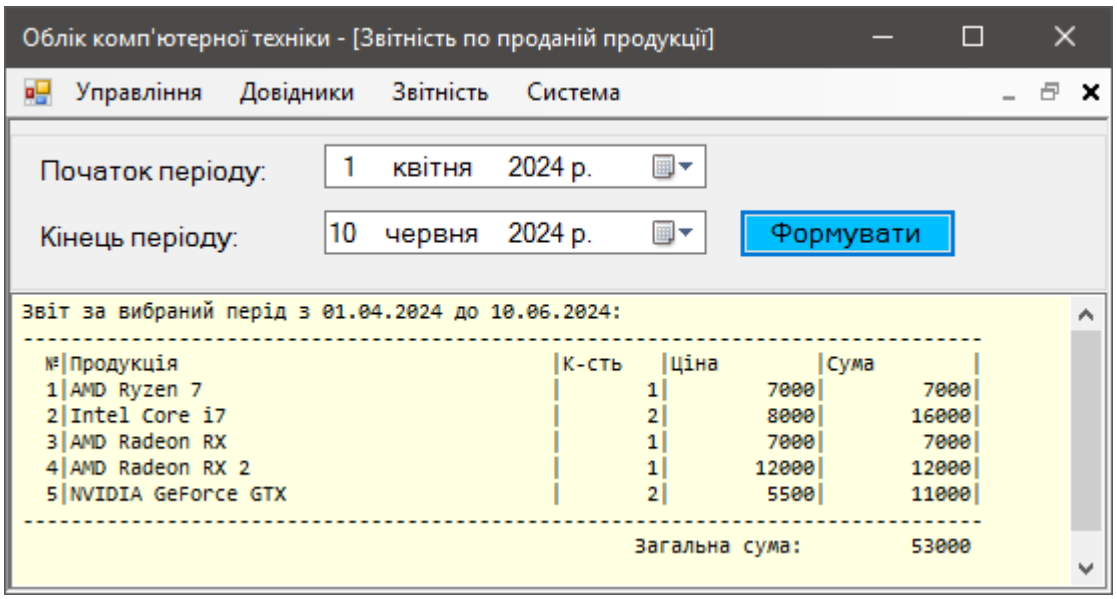


Рисунок 3.13 – Тестування форми «Звітність по проданій продукції»

Модульне тестування є ключовим етапом забезпечення якості програмного забезпечення, оскільки воно дозволяє виявити та виправити помилки на рівні окремих компонентів системи ще до їх інтеграції. Це забезпечує стабільну роботу кожного модуля та спрощує процес виявлення дефектів. На рис. 3.14 наведено результати модульного тестування.

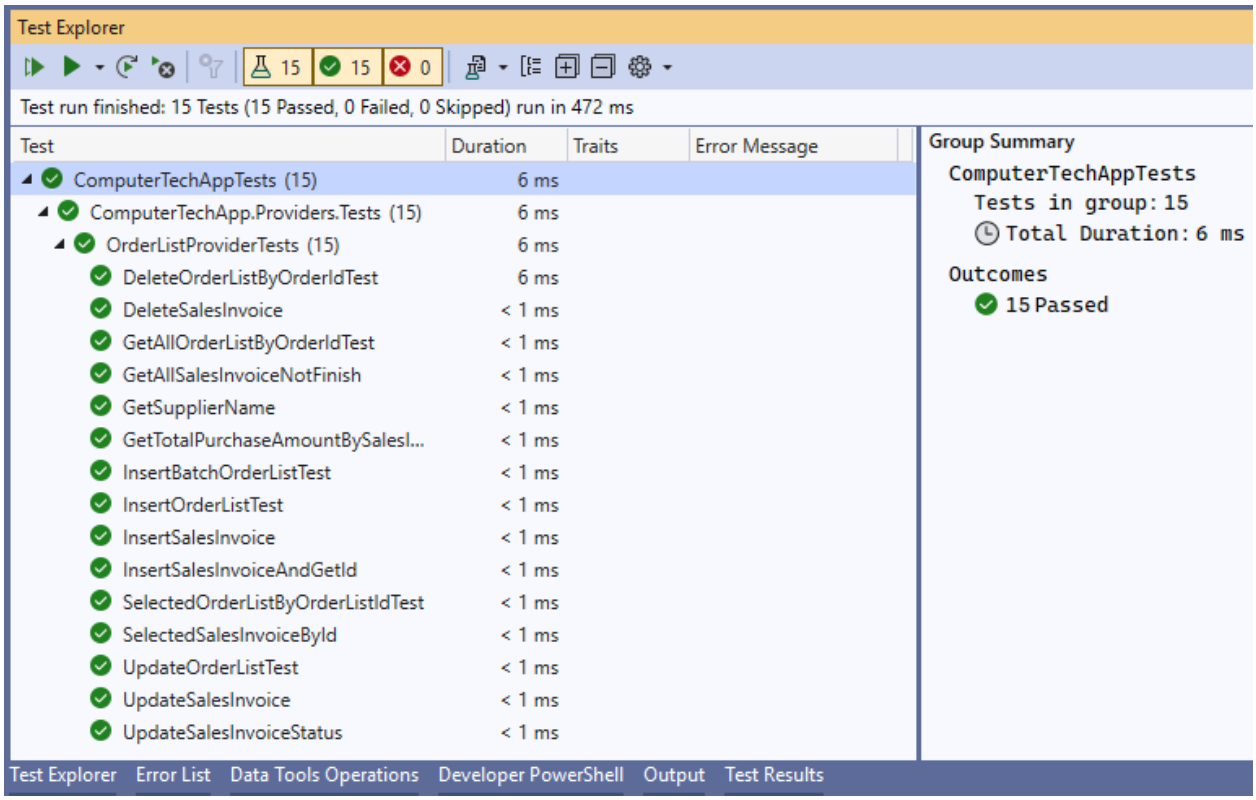


Рисунок 3.14 – Результати модульного тестування

Проведене тестування виявило, що всі основні модулі та функціональні елементи системи працюють коректно та відповідають заданим вимогам. Функціональне тестування підтвердило здатність системи виконувати необхідні бізнес-процеси, а модульне тестування забезпечило стабільність і надійність окремих компонентів. Виявлені незначні помилки були оперативно виправлені, що забезпечує високу якість програмного забезпечення. Загалом, система готова до впровадження та експлуатації в реальних умовах.

3.3 Інструкція користувача програми

Інструкція користувача програми призначена для ознайомлення з основними функціями та можливостями системи, а також надання допомоги під час її використанняНижче детально описані всі етапи взаємодії користувача з програмою, починаючи від авторизації та закінчуючи виконанням різних операцій. Першим кроком після запуску системи є авторизація користувача, яка забезпечує безпеку та персоналізований доступ до функцій програми. На рис. 3.15 наведено першу форму авторизації, з якою користувач буде взаємодіяти, вводячи свої облікові дані для входу в систему.

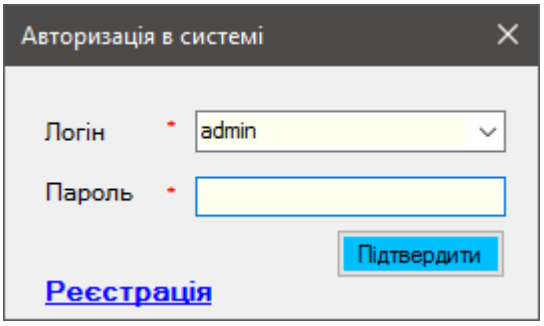


Рисунок 3.15 – Форма авторизації користувача

Користувач має ввести свої облікові дані, що складаються з логіну та пароля. Логін вибирається з випадаючого списку, в якому відображаються зареєстровані в системі користувачі, тоді як пароль вводиться вручну в текстове поле. Після введення необхідної інформації користувач натискає кнопку «Підтвердити» для завершення процесу авторизації. У разі відсутності облікового запису користувач може скористатися посиланням «Реєстрація» для створення нового облікового запису.

Користувач із роллю покупця, увійшовши до системи, може створювати накладні на купівлю комп'ютерної техніки (рис. 3.16). Для цього він обирає категорію товару з випадаючого списку та конкретний товар, який хоче придбати. Далі вводиться кількість одиниць товару, яку потрібно придбати. Користувач натискає кнопку «Додати», і обраний товар з'являється у таблиці праворуч, де відображаються всі поточні товари, додані до замовлення.

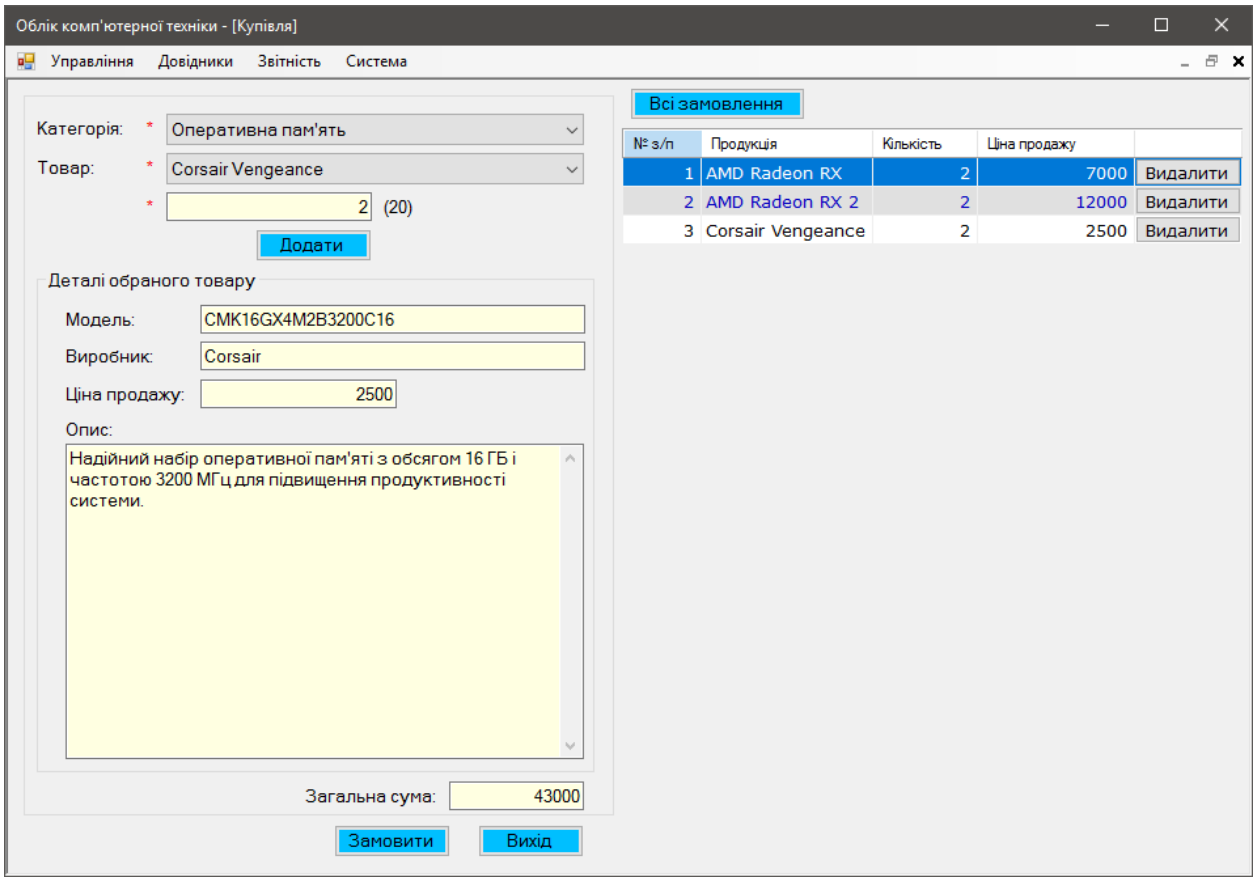


Рисунок 3.16 – Формування накладної

Форма також містить деталі обраного товару, такі як модель, виробник, ціна продажу та опис, що дозволяє користувачеві отримати повну інформацію про товар перед додаванням його до замовлення. Загальна сума замовлення автоматично підраховується та відображається у нижній частині форми. Після завершення формування замовлення користувач натискає кнопку «Замовити», щоб підтвердити та створити накладну.

Користувач із роллю покупця має можливість переглядати всю історію своїх покупок, натиснувши кнопку «Всі замовлення». Відкривається форма, в якій відображаються всі замовлення, зроблені користувачем, із зазначенням

дати замовлення, статусу виконання, загальної суми та переліку придбаних товарів. У таблиці праворуч можна побачити детальну інформацію про кожен товар у вибраному замовленні, включаючи кількість і ціну продажу. Сума кожного замовлення автоматично підраховується і відображається в нижній частині форми, разом із загальною сумою всіх покупок користувача. Такий інтерфейс забезпечує зручний доступ до історії замовлень та дозволяє користувачеві легко відслідковувати свої покупки. На рис. 3.17 наведено приклад форми, яка використовується для перегляду історії покупок.

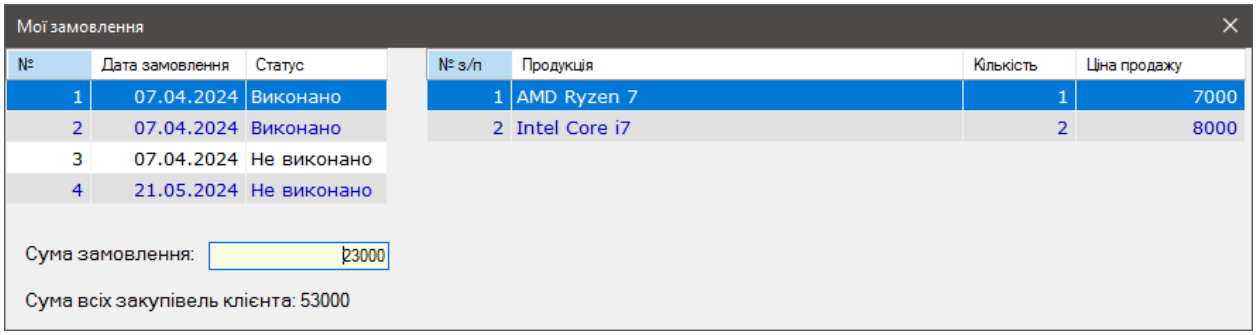


Рисунок 3.17 – Перегляд історії покупок

Користувач із роллю продавця має можливість підтвердити купівлю техніки, використовуючи спеціальну форму для підтвердження замовлень. У цій формі відображаються всі замовлення, які потребують підтвердження, із зазначенням дати замовлення, імені клієнта, та деталей замовлених товарів. Продавець може переглянути список товарів у кожному замовленні, включаючи їх кількість і ціну продажу. Після перевірки даних продавець вводить суму замовлення в відповідне поле та натискає кнопку «Підтвердити» для завершення процесу підтвердження купівлі. Крім того, у разі необхідності, продавець може видалити замовлення, натиснувши кнопку «Видалити». На рис. 3.18 наведено приклад форми, яка використовується продавцем для підтвердження купівлі техніки.

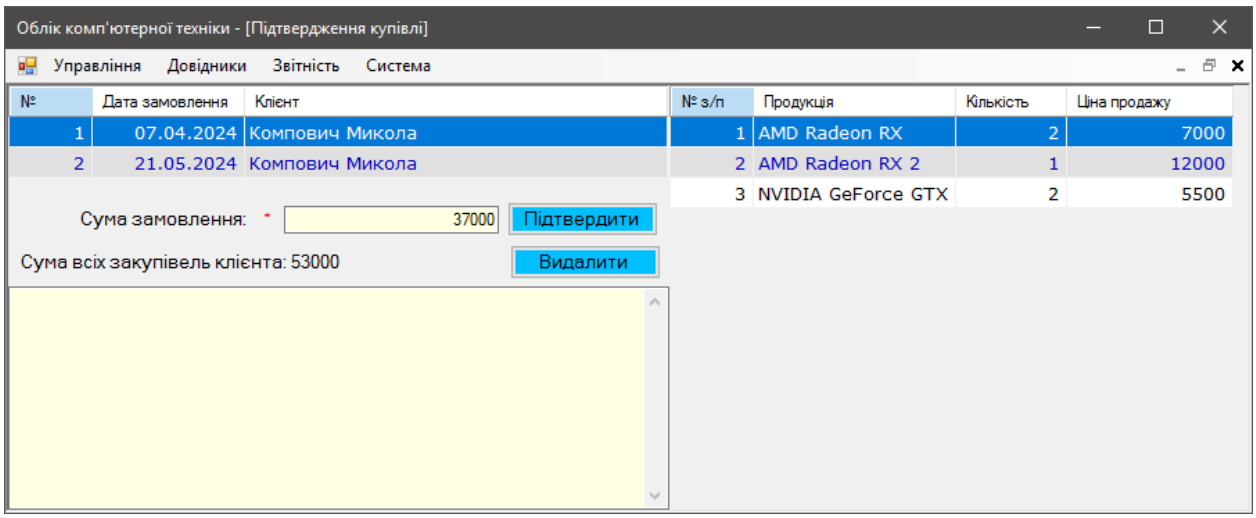


Рисунок 3.18 – Підтвердження купівлі техніки

Користувач із роллю продавця має можливість зробити замовлення у постачальників на постачання техніки, використовуючи спеціальну форму для створення замовлення (рис. 3.19). Продавець обирає категорію товару з випадаючого списку та конкретний товар, який потрібно замовити, вказуючи кількість одиниць, що потребуються. Натисканням кнопки «Додати» товар додається до списку замовлення, який відображається у таблиці праворуч.

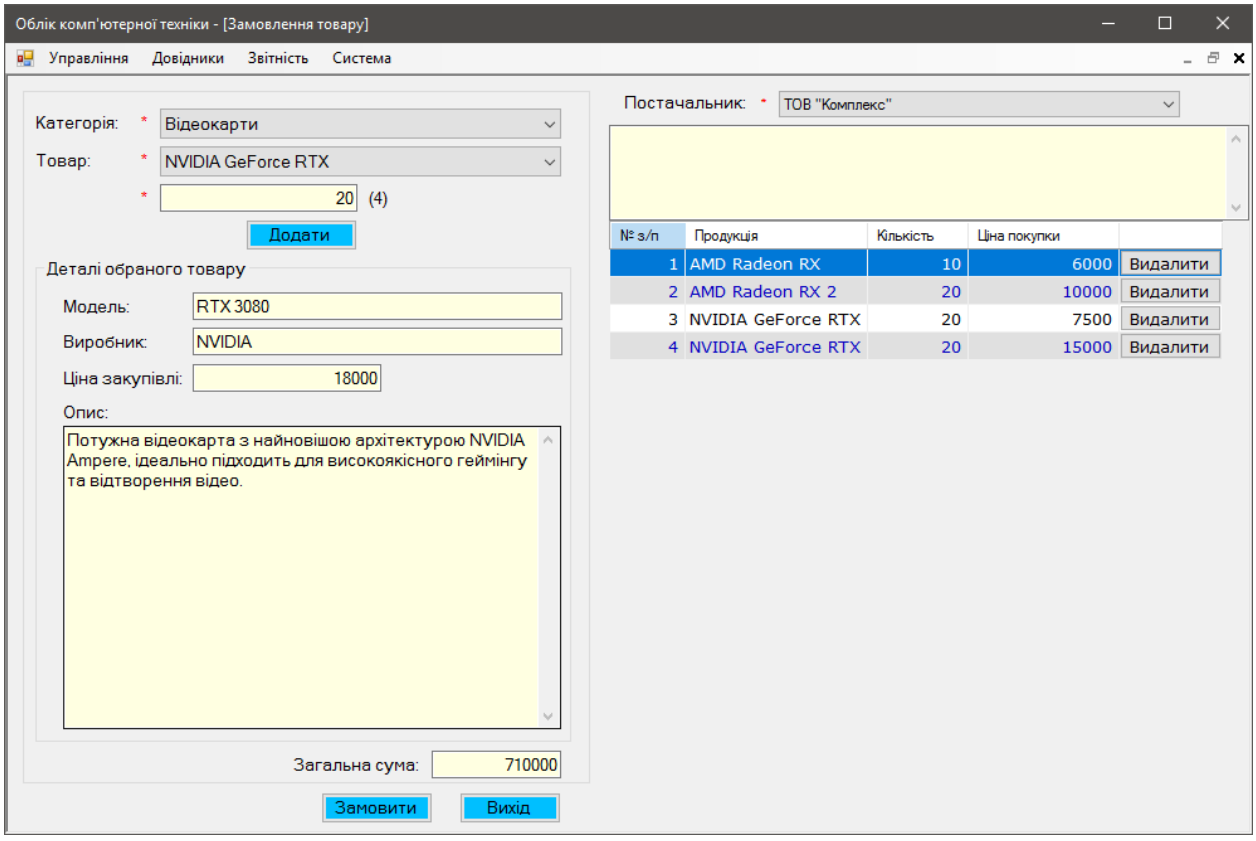


Рисунок 3.19 – Замовлення у постачальників на постачання техніки

Деталі обраного товару, такі як модель, виробник, ціна закупівлі та опис, автоматично заповнюються відповідними значеннями. Продавець також обирає постачальника з випадаючого списку постачальників. Загальна сума замовлення автоматично підраховується та відображається в нижній частині форми. Після завершення формування замовлення продавець натискає кнопку «Замовити» для підтвердження замовлення або «Вихід» для скасування операції.

Користувач із роллю робітника має можливість підтвердити надходження техніки на склад підприємства, використовуючи форму для обробки замовлень від постачальників (рис. 3.20). У цій формі відображаються всі замовлення, що потребують підтвердження, із зазначенням дати замовлення, постачальника та деталей замовлених товарів. Робітник може переглянути список товарів у кожному замовленні, включаючи їх кількість і ціну закупівлі.

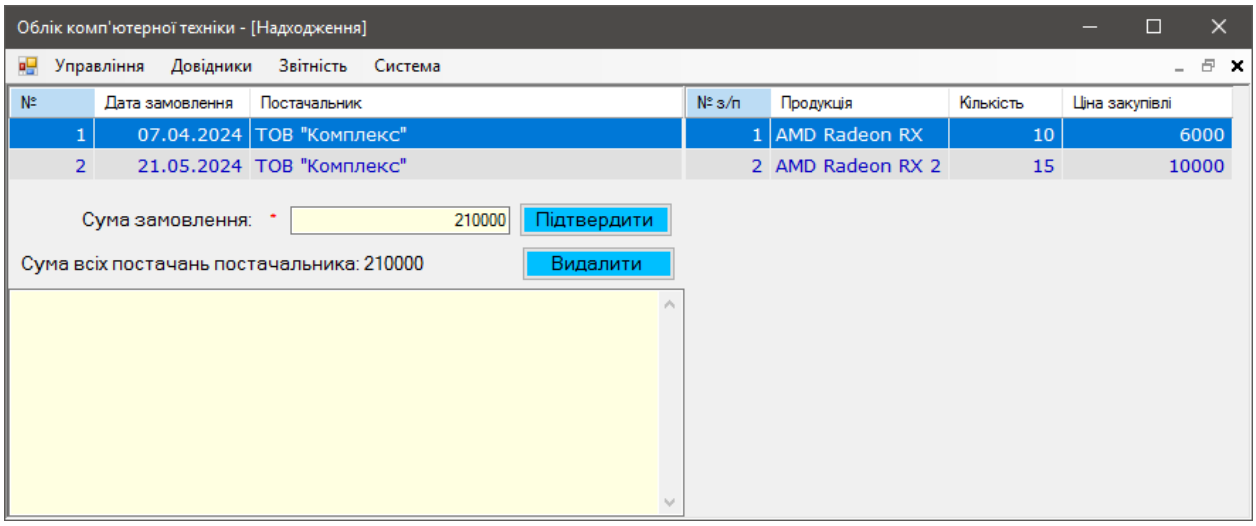


Рисунок 3.20 – Підтвердження надходження техніки на склад

Користувач із роллю робітника має можливість вести облік ремонту техніки за допомогою спеціальної форми, призначеної для реєстрації та управління ремонтними заявками. У цій формі робітник може обрати покупця з випадаючого списку та ввести назву пристрою, який потребує обслуговування. Також вказується дата обслуговування, тип обслуговування (наприклад, ремонт) та вартість робіт.

Після введення необхідної інформації робітник натискає кнопку «Додати», і пристрій додається до списку ремонтних заявок, який відображається у таблиці праворуч. У таблиці можна переглянути детальну інформацію про кожен зареєстрований ремонт, включаючи назву пристрою, дату обслуговування та вартість. За допомогою кнопок «Очистити» та «Вихід» робітник може скинути введені дані або вийти з форми відповідно. Такий інтерфейс забезпечує ефективне управління ремонтними заявками та дозволяє вести точний облік виконаних робіт. На рис. 3.21 наведено приклад форми, яка використовується робітником для ведення обліку ремонту техніки.

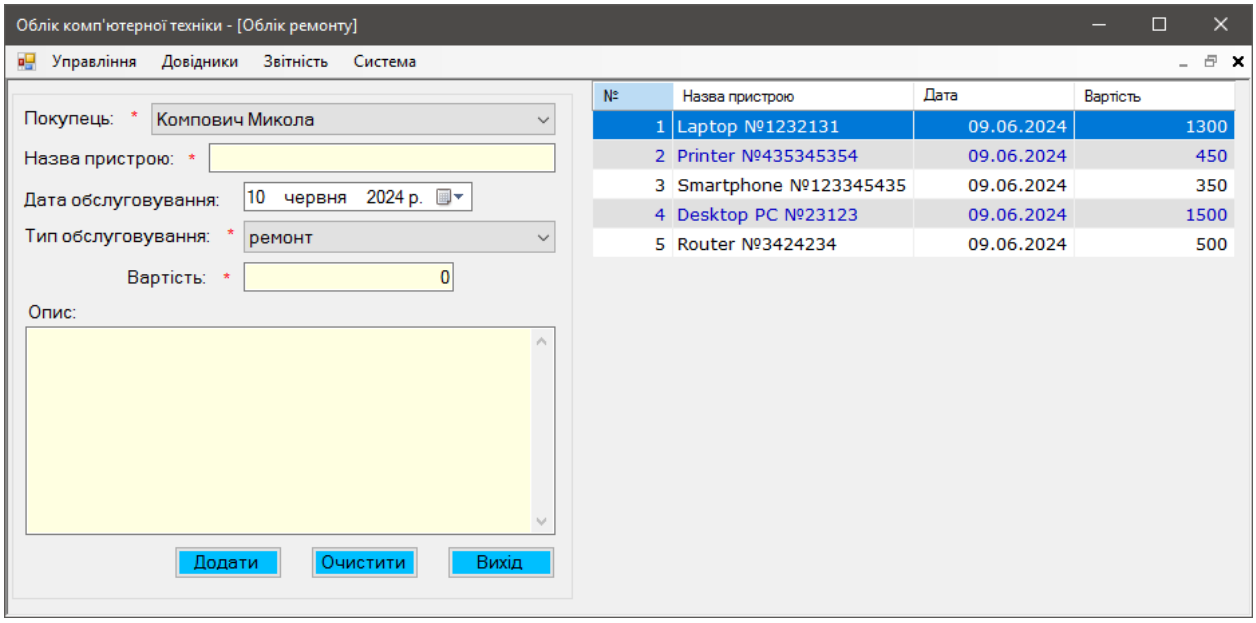


Рисунок 3.21 – Ведення обліку ремонту техніки

При натисканні на запис про техніку у списку ремонтних заявок, робітник має можливість змінити дані замовлення на ремонт, відкривши форму редагування. У цій формі відображаються всі деталі обраного замовлення, включаючи покупця, назву пристрою, дату обслуговування, тип обслуговування, вартість та опис робіт.

Робітник може змінити будь-які з цих даних, крім поля покупця, яке є заблокованим для редагування. Після внесення змін він може зберегти оновлену інформацію, натиснувши кнопку «Зберегти». Кнопка «Готово» дозволяє підтвердити завершення ремонту та закрити форму, зберігши внесені зміни. Якщо редагування потрібно скасувати, робітник може натиснути

кнопку «Вихід», щоб повернутися до попереднього вікна без збереження змін. Такий функціонал забезпечує гнучке управління даними про ремонтні замовлення та дозволяє робітнику оперативно вносити необхідні корективи. На рис. 3.22 наведено приклад форми редагування замовлення на ремонт.

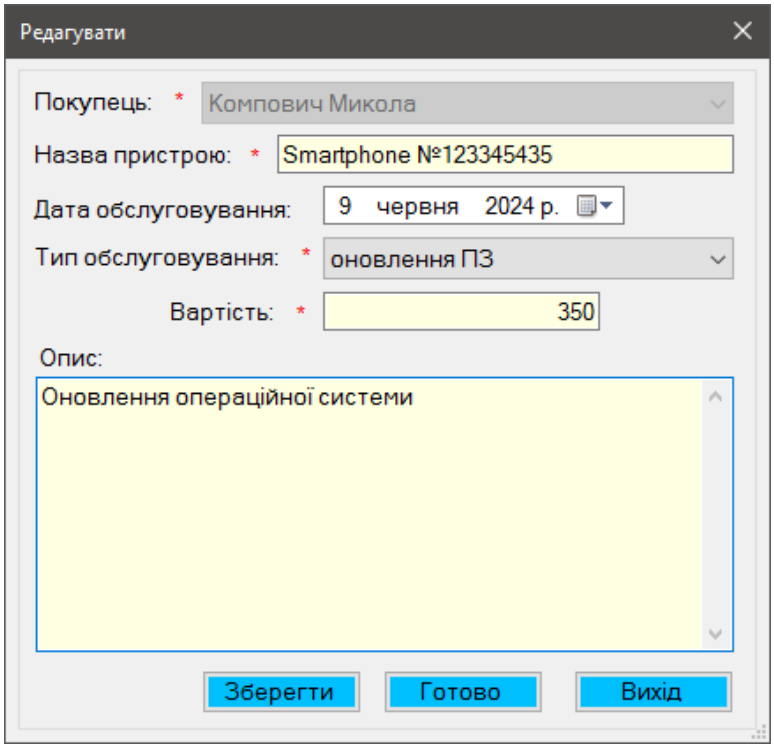


Рисунок 3.22 – Редагування замовлення на ремонт

Користувач із роллю директора має можливість вести облік комп'ютерної техніки за допомогою спеціальної форми. У цій формі директор може додавати нові товари, редагувати наявні та відстежувати їхні характеристики. Форма містить поля для введення категорії, назви, моделі, виробника, ціни продажу, кількості та опису товару. Всі обов'язкові поля позначені червоними зірочками.

Після заповнення необхідних даних директор натискає кнопку «Додати», щоб додати новий товар до бази даних. Усі додані товари відображаються в таблиці праворуч, де також зазначені їх назви, ціни продажу та кількість на складі. За допомогою кнопок «Очистити» та «Вихід» директор може скинути введені дані або вийти з форми відповідно. Такий інтерфейс забезпечує зручне управління обліком комп'ютерної техніки, дозволяючи директору ефективно контролювати наявність товарів та їх характеристики.

На рис. 3.23 наведено приклад форми, яка використовується директором для ведення обліку комп'ютерної техніки.

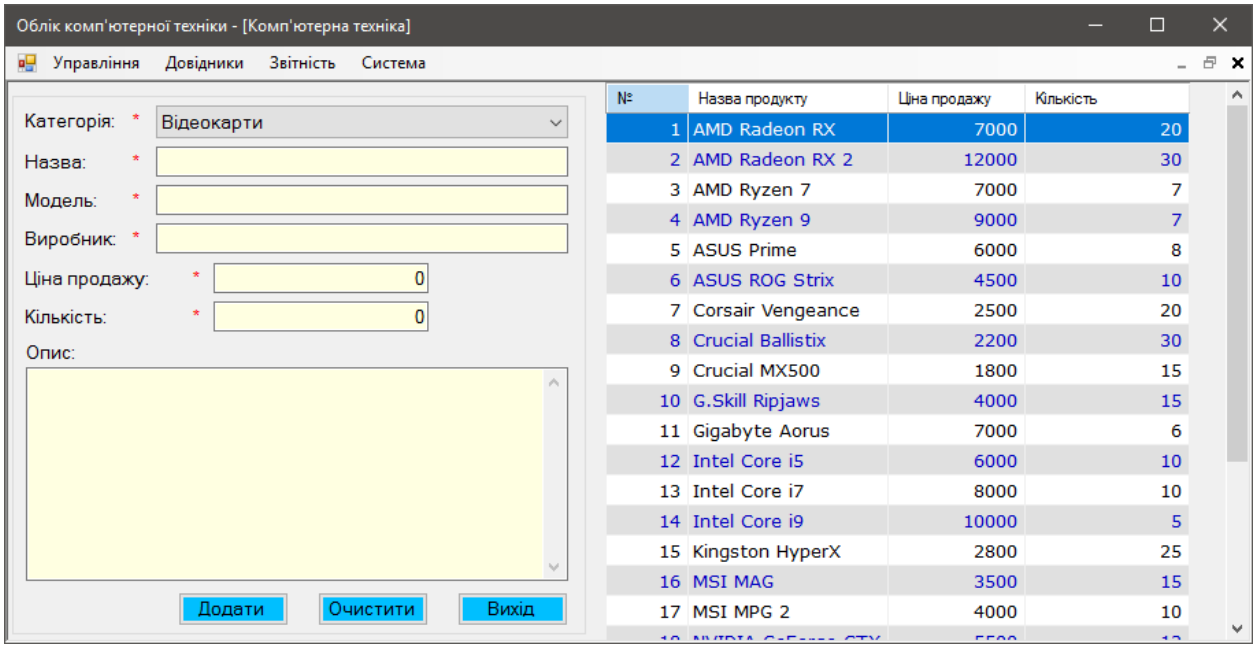


Рисунок 3.23 – Ведення обліку комп'ютерної техніки

Користувач із роллю директора має можливість формувати умови постачання для різних продуктів за допомогою спеціальної форми. У цій формі директор може вибрати постачальника з випадаючого списку та обрати продукцію, для якої необхідно встановити умови постачання. Далі вказуються мінімальний та максимальний об'єми постачання, ціна за одиницю продукції та умови доставки.

Після введення всіх необхідних даних директор натискає кнопку «Додати», щоб зберегти нові умови постачання. Усі створені умови відображаються в таблиці праворуч, де можна переглянути детальну інформацію про кожен продукт, його постачальника, мінімальний та максимальний об'єми постачання, а також ціну за одиницю. За допомогою кнопок «Очистити» та «Вихід» директор може скинути введені дані або вийти з форми відповідно. Такий інтерфейс дозволяє ефективно керувати умовами постачання, забезпечуючи належний контроль за постачанням продукції на підприємство. На рис. 3.24 наведено приклад форми, яка використовується директором для формування умов постачання.

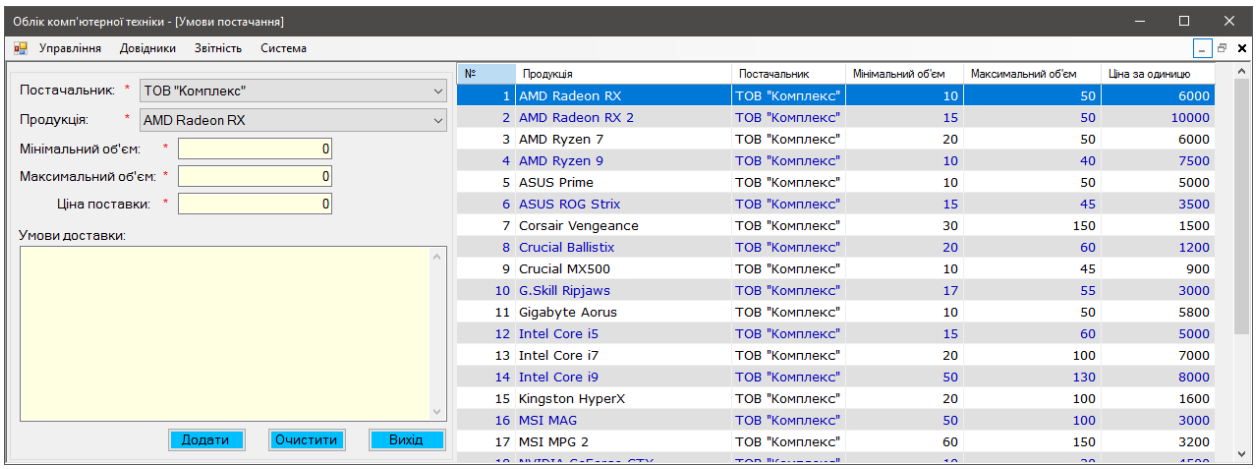


Рисунок 3.24 – Формування умов постачання

Користувач із роллю директора або робітника має можливість формувати звітність по проданій продукції за допомогою спеціальної форми. У цій формі користувач вказує початок та кінець періоду, за який необхідно сформувати звіт, обираючи відповідні дати з календаря. Після введення потрібних дат користувач натискає кнопку «Формувати», що генерує звіт про продану продукцію за вибраний період.

Звіт відображається у таблиці нижче, де зазначаються назви проданих продуктів, кількість проданих одиниць, ціна за одиницю та загальна сума продажу для кожного товару. Крім того, у нижній частині звіту відображається загальна сума продажу за весь період. Такий інтерфейс дозволяє користувачам ефективно відстежувати результати продажів, аналізувати успішність товарів та приймати відповідні управлінські рішення. На рис. 3.25 наведено приклад форми, яка використовується для формування звітності по проданій продукції.

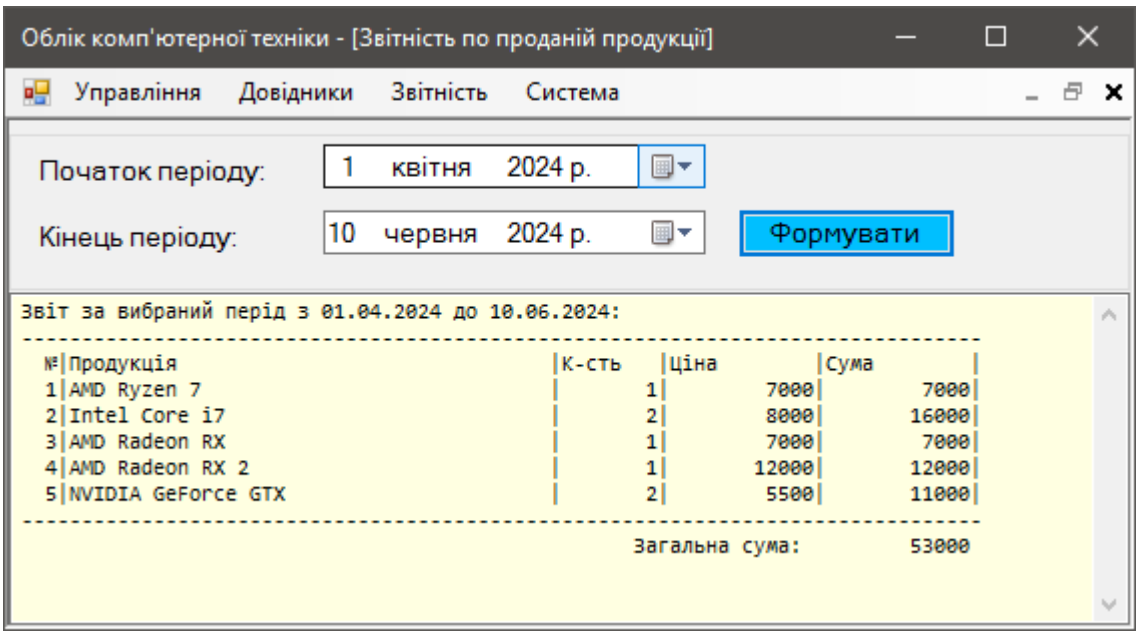


Рисунок 3.25 – Формування звітності по проданій продукції

Користувач із роллю директора або робітника має можливість формувати звітність по надходженню продукції за допомогою спеціальної форми. У цій формі користувач вказує початок та кінець періоду, за який необхідно сформувати звіт, обираючи відповідні дати з календаря. Після введення потрібних дат користувач натискає кнопку «Формувати», що генерує звіт про надходження продукції за вибраний період.

Звіт відображається у таблиці нижче, де зазначаються назви отриманих продуктів, кількість отриманих одиниць, ціна за одиницю та загальна сума для кожного товару. У нижній частині звіту відображається загальна сума надходжень за весь період. Такий інтерфейс дозволяє користувачам ефективно відстежувати надходження товарів, аналізувати обсяги поставок та приймати відповідні управлінські рішення. На рис. 3.26 наведено приклад форми, яка використовується для формування звітності по надходженню продукції.

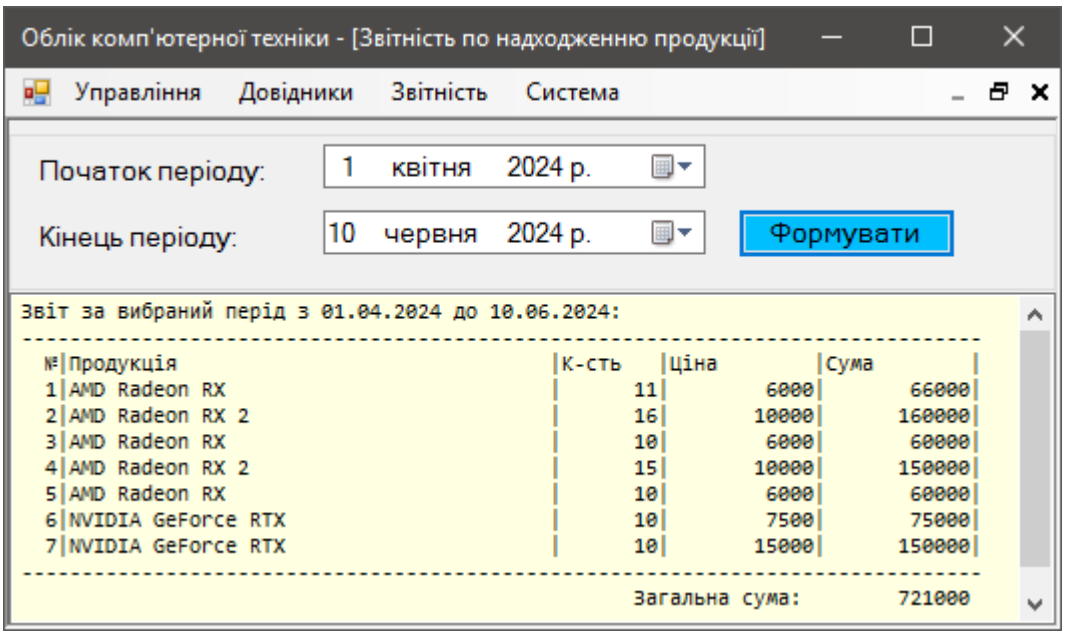


Рисунок 3.26 – Звітність по надходженню продукції

Ця форма забезпечує зручний інтерфейс для перегляду інформації про інструменти, видані за конкретний період, що дозволяє користувачам ефективно відстежувати та аналізувати використання інструментів у різні періоди часу. Це сприяє покращенню управління інструментами та оптимізації їх використання.

Користувач із роллю директора має можливість змінювати дані облікових записів та керувати ролями за допомогою спеціальної форми. У цій формі директор може додавати нових користувачів, редагувати інформацію про наявних користувачів та призначати їм відповідні ролі. Форма містить поля для введення прізвища, імені, опису, вибору ролі, а також логіну та паролю.

Після заповнення необхідних даних директор натискає кнопку «Додати», щоб створити новий обліковий запис або оновити існуючий. Усі користувачі відображаються в таблиці праворуч, де зазначені їх прізвище, ім'я, логін та роль. За допомогою кнопок «Очистити» та «Вихід» директор може скинути введені дані або вийти з форми відповідно. Такий інтерфейс забезпечує зручне управління користувачами системи, дозволяючи директору ефективно контролювати доступ до різних функцій програми. На рис. 3.27

наведено приклад форми, яка використовується директором для управління обліковими записами та ролями користувачів.

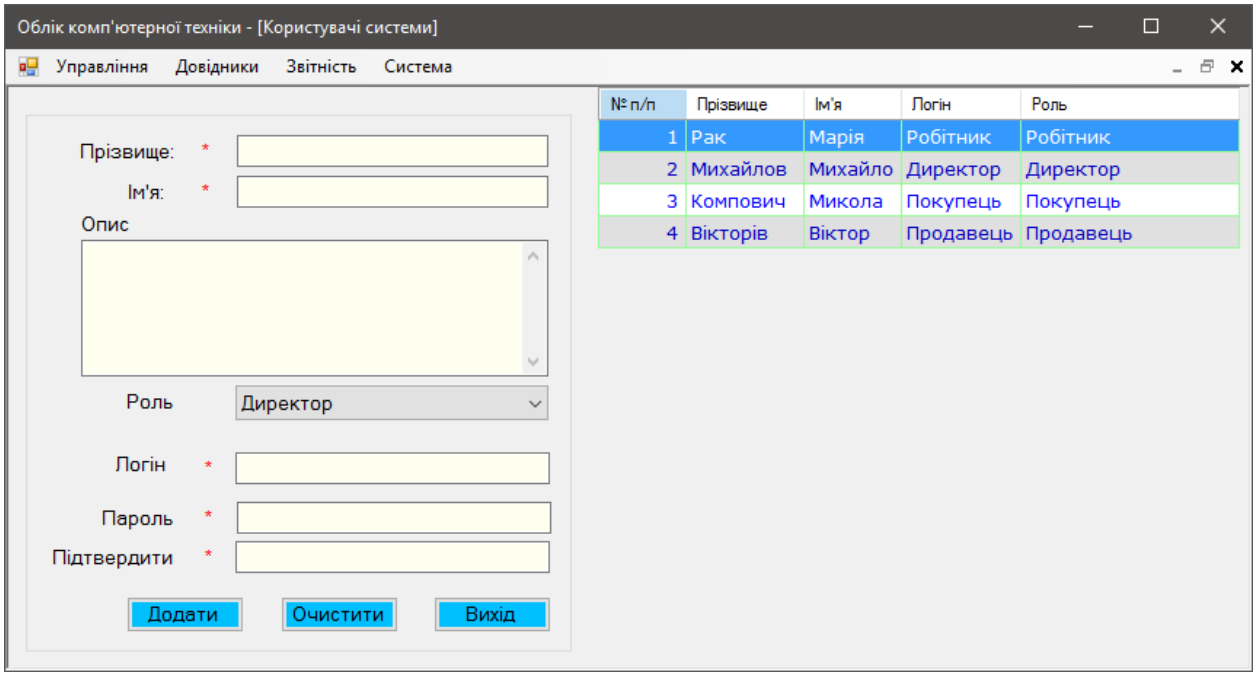


Рисунок 3.27 – Управління обліковими записами та ролями користувачів

Для виходу з програми користувач має перейти до меню «Управління» і обрати пункт «Вихід». Це забезпечує коректне завершення роботи з системою та збереження всіх внесених змін.

Таким чином, система для обліку комп'ютерної техніки забезпечує ефективне управління всіма аспектами діяльності підприємства, включаючи облік комп’ютерної техніки, формування звітності, управління користувачами та підтвердження надходжень. Розроблений інтерфейс є зручним і інтуїтивно зрозумілим, що сприяє підвищенню продуктивності праці та забезпечує належний контроль за всіма процесами.

3.4 Висновок

У рамках даного розділу було виконано детальну розробку модулів програмного забезпечення, включаючи опис створення та функціонування різних форм і методів системи. Було розроблено основні форми для обліку комп'ютерної техніки, замовлень, звітності, управління користувачами та інших важливих аспектів діяльності підприємства. Крім того, здійснено

функціональне та модульне тестування розробленого програмного забезпечення, що включало створення і виконання тестових сценаріїв для перевірки коректності роботи кожного модуля і системи в цілому. Це дозволило виявити та виправити помилки, забезпечивши високу якість та надійність програми.

Було розроблено інструкцію користувача, яка включає докладні описи використання кожної форми, а також приклади виконання основних операцій, таких як створення замовлень, облік товарів, формування звітності та управління обліковими записами. Це сприяє спрощенню процесу навчання нових користувачів та підвищенню ефективності їхньої роботи з системою.

ВИСНОВКИ

Дана робота присвячена розробці інформаційної системи для оптимізації діяльності підприємства «ТехноСервіс» з продажу та обслуговування комп'ютерної техніки. У системі реалізовано функціонал для ведення обліку комп'ютерної техніки, створення та обробки замовлень, формування звітності по проданій та отриманій продукції, управління користувачами та їх ролями, а також підтвердження надходжень та обліку ремонтів. Інтерфейс користувача забезпечує зручне та інтуїтивно зрозуміле виконання цих операцій, що сприяє підвищенню ефективності бізнес-процесів підприємства.

У першому розділі було здійснено аналіз організації та функціонування підприємства, досліджено існуючі моделі продаж та взаємодії з клієнтами, а також проведено аналіз програмних забезпечень для вирішення виявлених проблем. На основі проведеного аналізу було прийнято рішення про розробку власної системи, враховуючи специфічні потреби підприємства та ролі користувачів.

В другому розділі проведено вибір технологій для реалізації проекту, визначено функціональні та нефункціональні вимоги до системи, а також побудовано діаграми прецедентів для основних ролей користувачів. Моделювання основних бізнес-процесів, таких як додавання умов постачання продукції, формування накладної для купівлі комплектуючих та формування звітності, дозволило чітко окреслити ключові етапи роботи системи. Розроблено структуру бази даних, описано таблиці та побудовано фізичну модель бази даних. Обрано трирівневу архітектуру проекту, яка включає рівні презентації, логіки бізнесу та даних, що забезпечує ефективну роботу системи.

Третій розділ був присвячений розробці модулів програмного забезпечення, тестуванню та створенню інструкції користувача програми. Описано процес розробки форм та методів системи, проведено функціональне та модульне тестування з використанням тестових сценаріїв, що забезпечило

високу якість та надійність програмного забезпечення. Розроблена інструкція користувача допомагає користувачам швидко ознайомитися з системою та ефективно використовувати її функціональні можливості.

Проведена робота демонструє застосування сучасних методів розробки інформаційних систем для підвищення ефективності та якості управління підприємством. Розроблена система дозволяє автоматизувати бізнес-процеси, забезпечує точний облік продажів та обслуговування комп'ютерної техніки, покращує взаємодію з клієнтами та постачальниками. Отримані результати дозволять підприємству «ТехноСервіс» покращити свою конкурентоспроможність на ринку та забезпечити високий рівень обслуговування клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hendershott Terrence; Zhang Jie. A model of direct and intermediated sales. Journal of Economics & Management Strategy, 2006. – 316 p.
2. 9 Direct Sales Benefits (With Definition, Types and Tips) - Indeed URL: https://www.indeed.com/career-advice/career-development/direct-sales-benefits (дата звернення 10.06.2024).
3. The Ultimate Guide to Channel Sales URL: https://blog.hubspot.com/sales/channel-sales (дата звернення 10.06.2024).
4. What Is Channel Sales and What Are Its Pros and Cons? URL: https://taskdrive.com/sales/channel-sales/ (дата звернення 10.06.2024).
5. What is ecommerce? Definition, types, examples, benefits URL: https://printify.com/pod-glossary/ecommerce/ (дата звернення 10.06.2024).
6. What Is Ecommerce? | Benefits of the Ecommerce Model URL: https://www.the-future-of-commerce.com/2020/01/19/what-is-e-commerce-definition-examples/ (дата звернення 10.06.2024).
7. Bellu Renato. Microsoft Dynamics 365 for dummies. John Wiley & Sons, 2018. –94 p.
8. Newell Eric. Mastering Microsoft Dynamics 365 Implementations. John Wiley & Sons, 2021. –85 p.
9. What are the benefits of Microsoft Dynamics 365? URL: https://thecrmteam.com/what-are-the-benefits-of-microsoft-dynamics-365/ (дата звернення 10.06.2024).
10. What is Zoho CRM ? | Overview of Zoho CRM Software URL: https://www.zoho.com/crm/what-is-zoho-crm.html (дата звернення 10.06.2024).
11. Zoho CRM URL: https://www.zoho.com/crm/ (дата звернення 10.06.2024).
12. Benefits of Using Zoho CRM URL: https://uk.crmoz.com/blogs/post/advantages-using-zoho-crm (дата звернення 10.06.2024).

13. Odoo: Відкритий код ERP та CRM URL: https://www.odoo.com/uk_UA (дата звернення 10.06.2024).

14. Odoo CRM - Функції URL: https://www.odoo.com/uk_UA/app/ crm-features (дата звернення 10.06.2024).

15. Comprehensive Guide to Benefits of Odoo CRM URL: https://www.arrowhitech.com/market-insight/odoo-crm/ (дата звернення 10.06.2024).

16. Python URL : https://www.python.org/ (дата звернення 10.06.2024).

17. Java URL:https://dou.ua/lenta/articles/how-to-learn-java/ (дата звернення 10.06.2024).

18. C# URL: https://beetroot.academy/blog/courses/what-is-C (дата звернення 10.06.2024).

19. Гулковський М. М., Бурак Н. Є. Сучасні системи управління базами даних СУБД. 2020. – 43с.

20. Microsoft SQL Server URL: https://www.metabase.com/data_sources/ microsoft-sql-server (дата звернення 10.06.2024).

21. Vaswani Vikram. MySQL database usage & administration. McGraw Hill Professional, 2009. – 369 p.

22. Oracle Database URL: https://www.oracle.com/cis/database/ (дата звернення 10.06.2024).

23. BITTNER, Kurt; SPENCE, Ian. Use case modeling. Addison-Wesley Professional, 2003. – 86 p.

Додаток А. Лістинги програми

Лістинг 1. Код класу «ServiceRequestsProvider»

```
using ComputerTechApp.AppCode;
using MySqlConnector;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ComputerTechApp.Providers {
  internal class ServiceRequestsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings[«CONNECTSQL»];

    public void InsertServiceRequests(int UsersId, string DeviceName, DateTime ServiceDate,
string ServiceType, double Cost, string Description) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «INSERT INTO ServiceRequests (UsersId, DeviceName, ServiceDate,
ServiceType, Cost, Description, Status) « +
        «VALUES (@UsersId, @DeviceName, @ServiceDate, @ServiceType, @Cost,
@Description, @Status)»;
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue(«@UsersId», UsersId);
      cmd.Parameters.AddWithValue(«@DeviceName», DeviceName);
      cmd.Parameters.AddWithValue(«@ServiceDate», ServiceDate);
      cmd.Parameters.AddWithValue(«@ServiceType», ServiceType);
      cmd.Parameters.AddWithValue(«@Cost», Cost);
      cmd.Parameters.AddWithValue(«@Description», Description);
      cmd.Parameters.AddWithValue(«@Status», 0);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
    }

    public List<ServiceRequests> GetAllServiceRequests() {
      int i = 0;
      List<ServiceRequests> ServiceRequestsList = new List<ServiceRequests>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «SELECT * FROM ServiceRequests WHERE Status=0»;
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        ServiceRequests selectedRequest = new ServiceRequests();
        selectedRequest.Number = ++i;
        selectedRequest.ServiceRequestsId = Convert.ToInt32(reader[«ServiceRequestsId»]);
        selectedRequest.UsersId = Convert.ToInt32(reader[«UsersId»]);
        selectedRequest.DeviceName = reader[«DeviceName»].ToString();
        selectedRequest.ServiceDate = Convert.ToDateTime(reader[«ServiceDate»]);
```

```
        selectedRequest.ServiceType = reader[«ServiceType»].ToString();
        selectedRequest.Cost = Convert.ToDouble(reader[«Cost»]);
        selectedRequest.Description = reader[«Description»].ToString();
        selectedRequest.Status = Convert.ToByte(reader[«Status»]);
        ServiceRequestsList.Add(selectedRequest);
      }
      reader.Close();
      connection.Close();

      if (ServiceRequestsList.Count == 0) {
        ServiceRequests noRequest = new ServiceRequests();
        noRequest.ServiceRequestsId = 0;
        noRequest.Message = NamesMy.NoDataNames.NoDataInServiceRequests;
        ServiceRequestsList.Add(noRequest);
      }
      return ServiceRequestsList;
    }

    public ServiceRequests SelectedServiceRequestById(int ServiceRequestsId) {
      ServiceRequests selectedRequest = new ServiceRequests();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «SELECT * FROM ServiceRequests WHERE ServiceRequestsId=« +
ServiceRequestsId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedRequest.ServiceRequestsId = Convert.ToInt32(reader[«ServiceRequestsId»]);
        selectedRequest.UsersId = Convert.ToInt32(reader[«UsersId»]);
        selectedRequest.DeviceName = reader[«DeviceName»].ToString();
        selectedRequest.ServiceDate = Convert.ToDateTime(reader[«ServiceDate»]);
        selectedRequest.ServiceType = reader[«ServiceType»].ToString();
        selectedRequest.Cost = Convert.ToDouble(reader[«Cost»]);
        selectedRequest.Description = reader[«Description»].ToString();
        selectedRequest.Status = Convert.ToByte(reader[«Status»]);
      }
      reader.Close();
      connection.Close();
      return selectedRequest;
    }

    public void UpdateServiceRequest(int UsersId, string DeviceName, DateTime ServiceDate,
string ServiceType, double Cost, string Description, int ServiceRequestsId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «UPDATE ServiceRequests SET UsersId = @UsersId, DeviceName =
@DeviceName, ServiceDate = @ServiceDate, ServiceType = @ServiceType,» +
        « Cost = @Cost, Description = @Description « +
          «WHERE ServiceRequestsId = @ServiceRequestsId»;
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue(«@UsersId», UsersId);
      cmd.Parameters.AddWithValue(«@DeviceName», DeviceName);
      cmd.Parameters.AddWithValue(«@ServiceDate», ServiceDate);
```

```
      cmd.Parameters.AddWithValue(«@ServiceType», ServiceType);
      cmd.Parameters.AddWithValue(«@Cost», Cost);
      cmd.Parameters.AddWithValue(«@Description», Description);
      cmd.Parameters.AddWithValue(«@ServiceRequestsId», ServiceRequestsId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
    }

    public void UpdateServiceRequestStatus(int Status,  int ServiceRequestsId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «UPDATE ServiceRequests SET Status = @Status WHERE
ServiceRequestsId = @ServiceRequestsId»;
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue(«@Status», Status);
      cmd.Parameters.AddWithValue(«@ServiceRequestsId», ServiceRequestsId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
    }

    public void DeleteServiceRequest(int ServiceRequestsId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = «DELETE FROM ServiceRequests WHERE ServiceRequestsId =
@ServiceRequestsId»;
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue(«@ServiceRequestsId», ServiceRequestsId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
    }

  }
}


public class ServiceRequests {
  private int _Number; // Номер запиту
  private int _ServiceRequestsId; // Ідентифікатор запиту
  private int _UsersId; // Ідентифікатор користувача
  private string _DeviceName; // Назва пристрою
  private DateTime _ServiceDate; // Дата обслуговування
  private string _ServiceType; // Тип обслуговування (ремонт, технічне обслуговування,
оновлення програмного забезпечення)
  private double _Cost; // Вартість
  private string _Description; // Опис
  private byte _Status; // Статус
  private string _Message; // Повідомлення

  public ServiceRequests() {
    _Number = 0;
    _ServiceRequestsId = 0;
```

```
  _UsersId = 0;
  _DeviceName = string.Empty;
  _ServiceDate = DateTime.MinValue;
  _ServiceType = string.Empty;
  _Cost = 0.0;
  _Description = string.Empty;
  _Status = 0;
  _Message = string.Empty;
}

public int Number {
  set { _Number = value; }
  get { return _Number; }
}

public int ServiceRequestsId {
  set { _ServiceRequestsId = value; }
  get { return _ServiceRequestsId; }
}

public int UsersId {
  set { _UsersId = value; }
  get { return _UsersId; }
}

public string DeviceName {
  set { _DeviceName = value; }
  get { return _DeviceName; }
}

public DateTime ServiceDate {
  set { _ServiceDate = value; }
  get { return _ServiceDate; }
}

public string ServiceType {
  set { _ServiceType = value; }
  get { return _ServiceType; }
}

public double Cost {
  set { _Cost = value; }
  get { return _Cost; }
}

public string Description {
  set { _Description = value; }
  get { return _Description; }
}

public byte Status {
  set { _Status = value; }
```

```
    get { return _Status; }
   }

   public string Message {
    set { _Message = value; }
    get { return _Message; }
   }
}
```

Лістинг 2. Код класу «ServiceRequestsForm»

```
using ComputerTechApp.AppCode;
using ComputerTechApp.Forms.Dictionary;
using ComputerTechApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ComputerTechApp.Forms.Controls {
  public partial class ServiceRequestsForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ServiceRequestsProvider _ServiceRequestsPrivider = new ServiceRequestsProvider();
    private List<ServiceRequests> _ServiceRequestsList = new List<ServiceRequests>();
    private UsersProvider _UsersProvider = new UsersProvider();
    private List<Users> _UsersList = new List<Users>();


    public ServiceRequestsForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {

_ServiceRequestsPrivider.InsertServiceRequests(Convert.ToInt32(UsersCBox.SelectedValue),
DeviceNameTBox.Text, ServiceDateDTP.Value,
          ServiceTypeCBox.Text, Convert.ToDouble(CostTBox.Text), DescriptionTBox.Text);

        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
```

```
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void ServiceRequestsGridView_CellClick(object sender, DataGridViewCellEventArgs
e) {
      if (e.RowIndex >= 0 && ServiceRequestsGridView[0, e.RowIndex].Value.ToString()
        != _ServiceRequestsList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateServiceRequestsForm updateServiceRequestsForm =
          new UpdateServiceRequestsForm(Convert.ToInt32(ServiceRequestsGridView[0,
e.RowIndex].Value.ToString()));
        updateServiceRequestsForm.ShowDialog();
        DataLoad();
      }
    }

    private void LoadAllDate() {
      ServiceTypeCBox.SelectedIndex = 0;
      _UsersList = _UsersProvider.GetAllUsersCustomer();
      UsersCBox.DataSource = _UsersList;
      UsersCBox.ValueMember = «UsersId»;
      UsersCBox.DisplayMember = «FIO»;
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (ServiceRequestsGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ServiceRequestsGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ServiceRequestsList = _ServiceRequestsPrivider.GetAllServiceRequests();
        LoadDataInServiceRequestsGridView(_ServiceRequestsList);
        if (_selectedRowIndex == ServiceRequestsGridView.Rows.Count) {
          _selectedRowIndex = ServiceRequestsGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          ServiceRequestsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          ServiceRequestsGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInServiceRequestsGridView(List<ServiceRequests>
ServiceRequestsList) {
      ServiceRequestsGridView.DataSource = null;
      ServiceRequestsGridView.Columns.Clear();
      ServiceRequestsGridView.AutoGenerateColumns = false;
      ServiceRequestsGridView.RowHeadersVisible = false;
```

```
    ServiceRequestsGridView.DataSource = ServiceRequestsList;

    if (ServiceRequestsList.Count > 0) {
      if (ServiceRequestsList[0].Message ==
NamesMy.NoDataNames.NoDataInServiceRequests) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = «Message»;
        messageColumn.Width = ServiceRequestsGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        ServiceRequestsGridView.Columns.Add(messageColumn);
      } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = «ServiceRequestsId»;
        ServiceRequestsGridView.Columns.Add(DetailIdColumn);
        ServiceRequestsGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = «№ «;
        numberColumn.DataPropertyName = «Number»;
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        ServiceRequestsGridView.Columns.Add(numberColumn);

        DataGridViewColumn ServiceRequestsNameColumn = new
DataGridViewTextBoxColumn();
        ServiceRequestsNameColumn.HeaderText = «Назва пристрою»;
        ServiceRequestsNameColumn.DataPropertyName = «DeviceName»;
        ServiceRequestsNameColumn.Width = NamesMy.SizeOptins.NameSize;
        ServiceRequestsGridView.Columns.Add(ServiceRequestsNameColumn);

        DataGridViewColumn ServiceDateColumn = new DataGridViewTextBoxColumn();
        ServiceDateColumn.HeaderText = «Дата обслуговування»;
        ServiceDateColumn.DataPropertyName = «ServiceDate»;
        ServiceDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        ServiceDateColumn.Width = 120;
        ServiceRequestsGridView.Columns.Add(ServiceDateColumn);

        DataGridViewColumn CostColumn = new DataGridViewTextBoxColumn();
        CostColumn.HeaderText = «Вартість»;
        CostColumn.DataPropertyName = «Cost»;
        CostColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        CostColumn.Width = 120;
        ServiceRequestsGridView.Columns.Add(CostColumn);
      }
      for (int i = 0; i < ServiceRequestsGridView.Columns.Count; i++) {
        ServiceRequestsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
      }
    }
```

```
    }

    private void ClearAllControls() {
      DeviceNameTBox.Text = String.Empty;
      ServiceTypeCBox.SelectedIndex = 0;
      ServiceDateDTP.Value = DateTime.Now;
      DescriptionTBox.Text = String.Empty;
      CostTBox.Text = «0»;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(DeviceNameTBox.Text)) {
        DeviceNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        DeviceNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToDouble(CostTBox.Text)) {
        CostValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CostValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(UsersCBox.SelectedValue) > 0) {
        UsersValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UsersValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```

Лістинг 3. Код класу «UpdateServiceRequestsForm»

```
using ComputerTechApp.AppCode;
using ComputerTechApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ComputerTechApp.Forms.Controls {
  public partial class UpdateServiceRequestsForm : Form {
    private int _ServiceRequestsId = 0;
```

```
    private ServiceRequests _selectedServiceRequests = new ServiceRequests();
    private ServiceRequestsProvider _ServiceRequestsPrivider = new ServiceRequestsProvider();
    private ValidationMy _Validation = new ValidationMy();
    private UsersProvider _UsersProvider = new UsersProvider();
    private List<Users> _UsersList = new List<Users>();


    public UpdateServiceRequestsForm(int ServiceRequestsId) {
      InitializeComponent();
      _ServiceRequestsId = ServiceRequestsId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {

_ServiceRequestsPrivider.UpdateServiceRequest(Convert.ToInt32(UsersCBox.SelectedValue),
DeviceNameTBox.Text, ServiceDateDTP.Value,
          ServiceTypeCBox.Text, Convert.ToDouble(CostTBox.Text), DescriptionTBox.Text,
_ServiceRequestsId);
        this.Close();
      }
    }

    private void ChangeStatusBtn_Click(object sender, EventArgs e) {
       _ServiceRequestsPrivider.UpdateServiceRequestStatus(1, _ServiceRequestsId);
      MessageBox.Show(«Стан пристрою оновлено!»);
       this.Close();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _UsersList = _UsersProvider.GetAllUsersCustomer();
      UsersCBox.DataSource = _UsersList;
      UsersCBox.ValueMember = «UsersId»;
      UsersCBox.DisplayMember = «FIO»;

      _selectedServiceRequests =
_ServiceRequestsPrivider.SelectedServiceRequestById(_ServiceRequestsId);
      UsersCBox.SelectedValue = _selectedServiceRequests.UsersId;
      DeviceNameTBox.Text = _selectedServiceRequests.DeviceName;
      ServiceDateDTP.Value = _selectedServiceRequests.ServiceDate;
      ServiceTypeCBox.Text = _selectedServiceRequests.ServiceType;
      CostTBox.Text = _selectedServiceRequests.Cost.ToString();
      DescriptionTBox.Text = _selectedServiceRequests.Description;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
```

```
    if (_Validation.IsDataEntering(DeviceNameTBox.Text)) {
      DeviceNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      DeviceNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToDouble(CostTBox.Text)) {
      CostValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      CostValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (Convert.ToInt32(UsersCBox.SelectedValue) > 0) {
      UsersValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      UsersValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    return isCorrect;
  }

 }
}
```

Додаток Б. Скрипти бази даних

```
CREATE TABLE `servicerequests` (
  `ServiceRequestsId` int NOT NULL AUTO_INCREMENT,
  `UsersId` int DEFAULT NULL,
  `DeviceName` varchar(150) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `ServiceDate` date DEFAULT NULL,
  `ServiceType` varchar(45) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `Cost` decimal(12,2) DEFAULT NULL,
  `Description` text COLLATE utf8mb3_unicode_ci,
  `Status` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`ServiceRequestsId`),
  KEY `UsersId_fl4_idx` (`UsersId`),
  CONSTRAINT `UsersId_fl4` FOREIGN KEY (`UsersId`) REFERENCES `users` (`UsersId`)
)

CREATE TABLE `category` (
  `CategoryId` int NOT NULL AUTO_INCREMENT,
  `CategoryName` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `Description` text COLLATE utf8mb3_unicode_ci,
  PRIMARY KEY (`CategoryId`)
)

CREATE TABLE `logs` (
  `LogsId` int NOT NULL AUTO_INCREMENT,
  `UsersId` int DEFAULT NULL,
  `EventNameShow` text CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci,
  `EventDate` datetime DEFAULT NULL,
  PRIMARY KEY (`LogsId`),
  KEY `UsersId_fk_idx` (`UsersId`),
  CONSTRAINT `UsersId_fk` FOREIGN KEY (`UsersId`) REFERENCES `users` (`UsersId`)
)
```

```
CREATE TABLE `order_list` (
  `OrderListId` int NOT NULL AUTO_INCREMENT,
  `OrderId` int DEFAULT NULL,
  `ProductId` int DEFAULT NULL,
  `Quantity` int DEFAULT NULL,
  `SalePrice` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`OrderListId`),
  KEY `OrderId_fk_idx` (`OrderId`),
  KEY `ProductId_fk3_idx` (`ProductId`),
  CONSTRAINT `OrderId_fk` FOREIGN KEY (`OrderId`) REFERENCES `orders` (`OrderId`),
  CONSTRAINT `ProductId_fk3` FOREIGN KEY (`ProductId`) REFERENCES `products` (`ProductId`)
)

CREATE TABLE `orders` (
  `OrderId` int NOT NULL AUTO_INCREMENT,
  `UsersId` int NOT NULL,
  `OrderDate` date NOT NULL,
  `TotalPrice` decimal(12,2) DEFAULT NULL,
  `Status` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`OrderId`),
  KEY `UsersId_fk22_idx` (`UsersId`),
  CONSTRAINT `UsersId_fk22` FOREIGN KEY (`UsersId`) REFERENCES `users` (`UsersId`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci;


CREATE TABLE `products` (
  `ProductId` int NOT NULL AUTO_INCREMENT,
  `ProductName` varchar(255) COLLATE utf8mb3_unicode_ci NOT NULL,
  `Model` varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
```

```
`Manufacturer` varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
`SalePrice` decimal(12,2) DEFAULT NULL,
`Description` text CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci,
`Quantity` int NOT NULL,
`CategoryId` int NOT NULL,
PRIMARY KEY (`ProductId`),
KEY `CategoryId_fk_idx` (`CategoryId`),
CONSTRAINT `CategoryId_fk` FOREIGN KEY (`CategoryId`) REFERENCES `category` (`CategoryId`)
)

CREATE TABLE `products_list` (
`ProductsListId` int NOT NULL AUTO_INCREMENT,
`SalesInvoicesId` int NOT NULL,
`ProductId` int NOT NULL,
`Quantity` int NOT NULL,
`PurchasePrice` decimal(10,2) NOT NULL,
PRIMARY KEY (`ProductsListId`),
KEY `SalesInvoicesId_fk_idx` (`SalesInvoicesId`),
KEY `ProductId_fk4_idx` (`ProductId`),
CONSTRAINT `ProductId_fk4` FOREIGN KEY (`ProductId`) REFERENCES `products` (`ProductId`),
CONSTRAINT `SalesInvoicesId_fk` FOREIGN KEY (`SalesInvoicesId`) REFERENCES `salesinvoices` (`SalesInvoiceId`)
)

CREATE TABLE `salesinvoices` (
`SalesInvoiceId` int NOT NULL AUTO_INCREMENT,
`UsersId` int NOT NULL,
`InvoiceDate` date NOT NULL,
`TotalPrice` decimal(10,2) DEFAULT NULL,
`SupplierId` int DEFAULT NULL,
`Status` tinyint(1) DEFAULT NULL,
PRIMARY KEY (`SalesInvoiceId`),
```

```
    KEY `salesinvoices_ibfk_1_idx` (`UsersId`),
    KEY `SupplierId_fk3_idx` (`SupplierId`),
    CONSTRAINT `SupplierId_fk3` FOREIGN KEY (`SupplierId`) REFERENCES `suppliers` (`SupplierId`),
    CONSTRAINT `UsersId_fk5` FOREIGN KEY (`UsersId`) REFERENCES `users` (`UsersId`)
  )


  CREATE TABLE `suppliers` (
    `SupplierId` int NOT NULL AUTO_INCREMENT,
    `SupplierName` varchar(255) COLLATE utf8mb3_unicode_ci NOT NULL,
    `Address` varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    `Email` varchar(150) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci DEFAULT NULL,
    `Phone` varchar(45) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    PRIMARY KEY (`SupplierId`)
  )

  CREATE TABLE `supplyconditions` (
    `SupplyConditionId` int NOT NULL AUTO_INCREMENT,
    `SupplierId` int NOT NULL,
    `ProductId` int NOT NULL,
    `DeliveryTerms` varchar(1255) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci DEFAULT NULL,
    `MinVolume` int DEFAULT NULL,
    `MaxVolume` int DEFAULT NULL,
    `UnitPrice` decimal(10,2) NOT NULL,
    PRIMARY KEY (`SupplyConditionId`),
    KEY `SupplierId` (`SupplierId`),
    KEY `ProductId` (`ProductId`),
    CONSTRAINT `supplyconditions_ibfk_1` FOREIGN KEY (`SupplierId`) REFERENCES `suppliers` (`SupplierId`),
```

```
    CONSTRAINT `supplyconditions_ibfk_2` FOREIGN KEY (`ProductId`) REFERENCES `products` (`ProductId`)
  )

  CREATE TABLE `users` (
    `UsersId` int NOT NULL AUTO_INCREMENT,
    `FirstName` varchar(45) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    `LastName` varchar(45) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    `UsersName` varchar(45) CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    `UsersPassword` varchar(150) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
    `RoleId` int DEFAULT NULL,
    `Description` text CHARACTER SET utf8mb3 COLLATE utf8mb3_unicode_ci,
    PRIMARY KEY (`UsersId`)
  )
```