Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Центр післядипломної освіти

(повна назва)

Кафедра Програмної інженерії

(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для метрологічного обліку приладів

(тема)

Виконав:
студент 4 курсу, групи ПЗПІп-22-1
Мазуров М.М.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)
Керівник доц. кафедри ПІ Русакова Н.Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В. Дудар
(підпис) (посада, прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програми Програмна інженерія
(повна назва)

ЗАТВЕРЖДУЮ:
Зав. кафедри
_____
(підпис)
«___»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Мазурову Миколі Миколайовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмна система для метрологічного обліку приладів* затверджена наказом університету від "17" червня 2024 р. № 588 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22.07.2024 р.

3. Вихідні данні до роботи *вимоги до розроблюваної програми, вимоги до архітектури системи, електронні ресурси за обраною темою, мови програмування C#, технологія доступу до бази даних ADO.NET, СКБД MSSQL, середовище розробки Visual Studio.*

4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної галузі, формування вимог до програмного забезпечення, архітектура та проектування розробленого програмного забезпечення, опис прийнятих програмних рішень, тестування програмного забезпечення, висновки, додатки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Позначка про виконання
1	Аналіз проблемної області	6.05.2024 – 10.05.2024	Виконано
2	Розробка постановки задачі	10.05.2024 – 13.05.2024	Виконано
3	Аналіз та моделювання предметної області	13.05.2024 – 15.05.2024	Виконано
4	Проектування БД	15.05.2024 – 18.05.2024	Виконано
5	Розробка алгоритмів	18.05.2024 – 23.05.2024	Виконано
6	Проектування архітектури програмної системи	23.05.2024 – 25.05.2024	Виконано
7	Програмна реалізація системи	25.05.2024 – 12.06.2024	Виконано
8	Підготовка пояснювальної записки	12.06.2024 – 30.06.2024	Виконано
9	Підготовка презентації та доповіді	1.07.2024 – 07.07.2024	Виконано
10	Нормоконтроль	9.07.2024	Виконано
11	Рецензування	11.07.2024	Виконано
12	Занесення диплома в електронний архів	13.07.2024	Виконано
13	Попередній захист	15.07.2024	Виконано
14	Допуск до захисту у зав. кафедри	18.07.2024	Виконано

Дата видачі завдання 6 травня 2024р.

Студент _____ Мазуров М.М.

(підпис) (прізвище, ініціали)

Керівник роботи _____ доц.кафедри ПІ Русакова Н.Є.

(підпис) (прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра містить: 77 сторінок, 43 рисунки, 3 додатки, 13 джерел.

ДІАГРАМА, МЕТРОЛОГІЧНИЙ ПРИЛАД, ПРОГРАМНА СИСТЕМА, РЕЛЯЦІЙНА БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, ADO.NET, MICROSOFT SQL SERVER.

Об’єкт розробки – програмної системи метрологічного обліку приладів.

Мета розробки – проектування та розробка програмної системи, що забезпечує підтримку метрологічного обліку приладів на основних етапах від прийому заявки до виписки сертифікату.

Метод рішення – середовище розробки Microsoft Visual Studio 2022, платформа ASP.Net, мова програмування C#, інтерфейс програмування додатків Windows Forms, система управління базами даних Microsoft SQL Server.

В результаті спроектовано програмну систему, що складається з серверної та клієнтської частини та забезпечує підтримку метрологічного обліку приладів на основних етапах.

ADO.NET, DATABASE MANAGEMENT SYSTEM, DIAGRAM, METROLOGICAL DEVICE, MICROSOFT SQL SERVER, RELATIONAL DATABASE, SOFTWARE SYSTEM.

The object of development is a software system for metrological accounting of devices.

The goal of the development is the design and development of a software system that provides support for metrological accounting of devices at the main stages from receiving an application to issuing a certificate.

Solution method – Microsoft Visual Studio 2022 development environment, ASP.Net platform, C# programming language, Windows Forms application programming interface, Microsoft SQL Server database management system.

As a result, a software system was designed, consisting of a server and a client part and providing support for metrological accounting of devices at the main stages.

Я, Мазуров Микола Миколайович, студент гр. ПЗПІп-22-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для метрологічного обліку приладів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ЗР ЗВТ – законодавчо регульований засіб вимірювальної техніки;

КО – контактна особа;

НФ – нормальна форма;

МП – метрологічний прилад;

СКБД – система керування базою даних.

ВСТУП

В нашому часі складно уявити собі якісь галузі та процеси виробництва без використанні комп’ютерів та інших засобів обчислювальної техніки. Для того, щоб забезпечити єдність вимірювань на виробництві, в медицині, наукових закладах та багатьох інших галузях. в Україні та за її межами, метрологічні підрозділи проводять повірку засобів вимірювальної техніки.

Розробка програмної системи для метрологічного обліку приладів вимірювальної техніки є актуальною та дозволить значно оптимізувати та скоротити витрати часу для оформлення та супроводу приладів при проведенні їх метрологічної повірки.

Мета розробки є проектування та розробка програмної системи, що забезпечує підтримку метрологічного обліку приладів на основних етапах від прийому заявки до виписки сертифікату.

Під час написання кваліфікаційної роботи було проведено аналіз та моделювання предметної області метрологічного обліку приладів, спроектовано базу даних, розроблено архітектуру системи; спроектовано логіку серверної частини системи; програмно реалізовано базу даних та саму системи для метрологічного обліку приладів.

В результаті розроблено програмну систему, яка дозволить легко проводити облік замовників та їх замовлень, в автоматичному режимі формувати рахунок-фактуру для оплати замовлення та свідоцтво про повірку конкретного приладу вимірювальної техніки, тощо.

Додаток є інтуїтивно зрозумілим та може бути використаним персоналом підприємств Облстандартметрології, який має навички роботи з системою Microsoft Windows.

При розробці реляційної бази даних було використано систему керування базами даних Microsoft SQL Server. Для програмної реалізації використано середовище розробки Microsoft Visual Studio 2022, мова C#, інтерфейс Windows Forms.

Робота перевірена на унікальність тексту в мережі інтернет та базі ХНУРЕ (див. додаток А) та відповідає вимогам державних стандартів з оформлення. За результатами кваліфікаційної роботи було розроблено презентацію (див. додаток Б). Лістинг програмного коду наведено в додатку В.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Взаємопов’язана робота підприємств в усіх галузях виробництва можлива на сьогодні завдяки наявності стандартів та використанню спільних одиниць вимірювання тих чи інших показників фізичного світу. Для того, щоб забезпечити єдність вимірювань на виробництві, в медицині, наукових закладах та багатьох інших галузях. в Україні та за її межами, метрологічні підрозділи проводять повірку засобів вимірювальної техніки [1].

Розробка програмної системи для метрологічного обліку приладів вимірювальної техніки є актуальною та дозволить значно скоротити витрати часу для оформлення та супроводу приладів при проведенні їх метрологічної повірки.

Спочатку облік процесу повірки на усіх етапах виконувався паперовим чином, потім шліхом зберігання окремих документів електронним образом. Але на сьогодні кількість документів, що процесів повірок та обслуговування заявок від підприємств та організацій сягає сотень та тисяч документів, що потребує збереження подібної інформаціє в базах даних та відповідної її обробки за допомогою спеціальних програмних систем.

База даних (БД) - сукупність даних, яка організована спеціальним чином та зберігається за допомогою комп’ютерної техніки. Процес проектування баз даних складається з декількох етапів [2]. Першим етапом є аналіз та концептуальне моделювання предметної області, на яких описуються бізнес-процеси, що відбуваються в реальному світі; моделюються основні об’єкти та взаємовідносини між ними. На цьому етапі активно використовують UML-діаграми [3]. Наприклад, діаграма прецедентів дозволяє представити функціонал, який повинна надавати програмна система. Діаграма класів під час проектування БД дозволяє описати основні об’єктів, що фігурують в предметні області, як вони взаємодіють між собою.

На етапі інфологічного моделювання [2] будуються ER (від англ. Entity Relationship – “сутність-зв’язок”) діаграми [4], які надають більш формалізований опис сутностей БД та є основою для наступного етапу проектування БД.

Логічне моделювання БД полягає в створенні логічної структури БД для обраної логічної моделі. На сьогодні існує доволі багато видів баз даних [5]. Розглянемо найбільш популярні з них:

– реляційні БД зберігають дані у вигляді спеціальним чином організованих таблиць (relations) та зв’язків між ними;
– об’єктно-орієнтовані БД зберігають дані у вигляді об’єктів та забезпечують під час роботи з ними основні принципи об’єктно-орієнтованого підходу, типу наслідування, інкапсуляції та полімовфізму;
– NoSQL-системи складають окремий напрямок в області БД, які дозволяють зберігати дані за допомогою не жорстких структур (їх називають ще безсхемні); такі моделі використовуються щоб забезпечити високу продуктивність та масштабованість БД та програмних систем.

Доволі актуальними на практиці на сьогодні залишаються реляційні БД, що забезпечують простий процес їх проектування, надійність зберігання даних, зручний механізм роботи з ними за допомогою мови SQL [6].

Останнім етапом проектування БД є їх фізичного моделювання, коли на основі є логічної моделі бази даних реалізується реальна БД з використанням особливостей певної системи керування базами даних (СКБД). Різниця фізичних моделей буде полягати в специфічних типах даних, можливостях виділення пам’яті під об’єкти БД, тощо.

Важливим етапом розробки програмної системи є розробка її архітектури. Від її вибору залежить швидкість і вартість розробки, майбутня спроможність розширення системи, а також зручність її тестування. Найбільш

поширеними видами архітектури є клієнт-серверна, багатошарова, багаторівнева та мікросервісна архітектури.

Клієнт-серверна архітектура дозволяє рознести бізнес-логіку системи на рівні її частини. В випадку використання БД на окрему частину, сервер БД, виноситься не тільки робота самої СКБД, але і таблиці БД, і певна логіка системи, що може бути реалізована у вигляді тригерів чи збережених процедур та функцій.

Для підтримки будь якої бізнес-логіки програмної системи важливим є її аналіз та моделювання. Для підтримки обліку метрологічних приладів – це не є виключенням. Зручним та популярним інструментом моделювання є мова UML [3], важливими для розробки діаграмами в якій є діаграми класів, use-case діаграми, діаграми станів, тощо. Діаграми станів– це один з п'яти видів діаграм UML, призначених для моделювання динамічних аспектів поведінки систем. Діаграми станів підходять для моделювання життєвого циклу об'єкта. Діаграми стану застосовуються для моделювання динамічних аспектів поведінки систем (частіше це моделювання поведінки реактивних об'єктів.

Для розробників застосунків, що працюють з БД, важливою є DFD — діаграми потоків даних (Data Flow Diagrams), що використовуються для моделювання процесів переміщення даних у системі шляхом зображення мережевої структури даних. Зокрема, вони:

- дозволяють уявити систему з погляду даних;
- ілюструють зовнішні механізми подачі даних, які вимагатимуть наявності інтерфейсу певного виду;
- дозволяють уявити як автоматизовані, а й ручні процеси системи;
- виконують орієнтоване на дані секціонування всієї системи.

1.2 Аналіз існуючих аналогів

Під час роботи було проведено аналіз українського ринку подібних систем. У ході аналізу з’ясовано, що системи подібного типу є суто корпоративними і не доступні загалу. Тому було вирішено взяти функціональні

аналоги, які пов’язані з обліком. Отже, на сьогодні існує декілька програмних систем для обліку товарів зі схожим функціоналом, які реально використовуються в Україні. Серед проаналізованих програмних систем відмітимо такі системи, як «BAS Управління торгівлею» [7] та «SUBTOTAL» [8].

Програмна система «BAS Управління торгівлею» є набором інструментів, що дозволяє взаємодіяти з віддаленим сервером з базою даних, який надає можливість маніпуляції даними про товари (можемо розглядати їх як аналоги замовлень) та їх відображення. Особливістю системи є розташування серверів компанії у багатьох європейських країнах та забезпечення безпеки даних на високому рівні.

На рисунку 1 наведено головна форма системи «BAS Управління торгівлею».

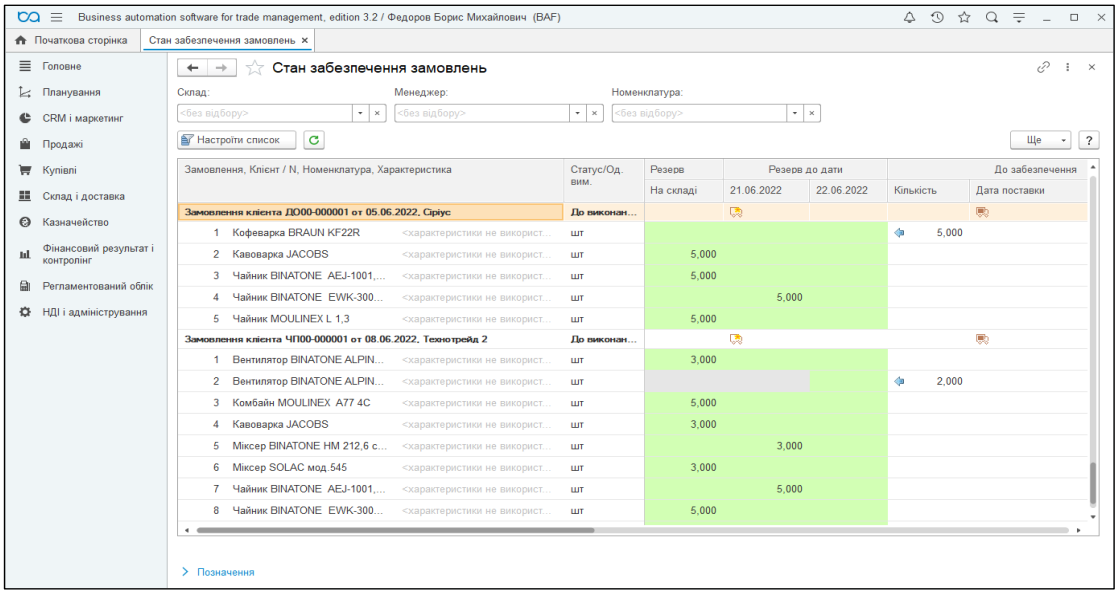


Рисунок 1 – Форма системи «BAS Управління торгівлею» (за даними [7])

Слід відмітити наступні переваги системи:

– планування та аналіз продажів, закупівель, а також підтримка переходів від одного стану закупівель до іншого;

- значний функціонал з управління продажами та поставками продуктів, запасами на складах;
- облік та аналіз грошових витрат;
- аналіз та формування звітності стосовно торгової діяльності;
- документація в системі формується відповідно до чинного законодавством України (це також робить систему аналогом);
- інтегрована робота з системами “BAS Бухгалтерія” та “BAS Роздрібна торгівля”.

Як недоліки слід зазначити:

- високу вартість програмної системи;
- відсутність веб-застосунку;
- відсутність англійської локалізації.

Програмна система «SUBTOTAL» являє собою зручний сервіс для зберігання різного типу даних та засобів взаємодії з ними. Програмна система позиціонується як система з можливістю інтеграції з різними інтернет-сервісами або інтернет-магазинами та ін.

Зображення однієї з форми застосунку наведено на рисунку 2.

До переваг даної програмної системи слід віднести:

- інтеграцію з інтернет-магазинами, бухгалтерією, банками та онлайн касами;
- розвинутий функціонал управління складами, в тому числі, контроль залишків, оцінка товарів, підтримка документообігу;
- підтримка бази постачальників;
- зручний дружній інтерфейс;
- можливість доступу до системи зі смартфону, ПК або ноутбука.

До недоліків системи можна віднести:

- відсутність української локалізації, що важливо для нашої системи;
- обмежений функціонал з точки зору довідникових таблиць;
- система зараз недоступна в Україні.

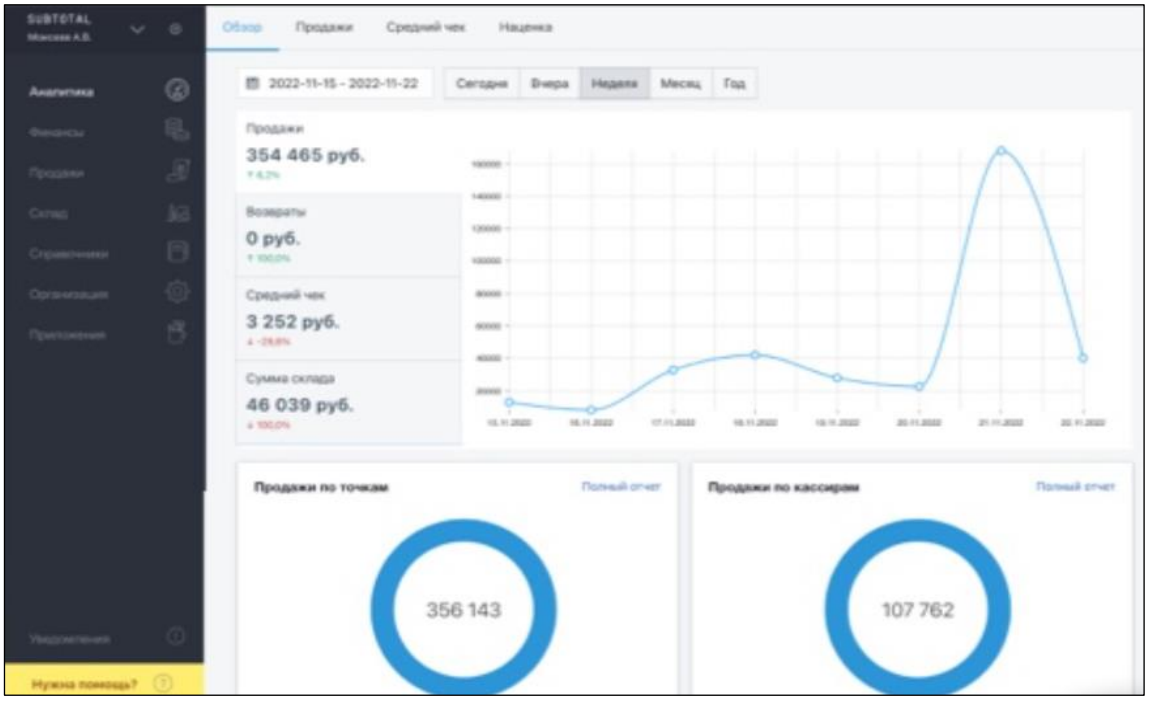


Рисунок 2 – Сервіс «SUBTOTAL» (за даними [8])

Проведений аналіз аналогів дозволив сформулювати більш чіткі вимоги під час постановки задачі на розробку власної програмної системи.

1.3 Постановка задачі

Необхідно розробити програмну системи, яка буде складатися з серверної та клієнтської частин, та буде забезпечувати підтримку метрологічного обліку законодавчо регульованих засобів вимірювальної техніки (ЗР ЗВТ) (або простіше метрологічних приладів (МП)) на основних етапах від прийому заявки до виписки сертифікату. Система повинна забезпечувати виконання наступних функцій:

- система повинна відображати дані про основні об’єкти предметної області: замовників, виконавців, прилади, замовлення, рахунок-фактура, категорії приладів;
- система повинна підтримувати арифметичну обробку даних стосовно строків проведення повірки ЗР ЗВТ, розрахунку вартості робіт при оформленні замовлення, тощо;

- система повинна підтримувати сортування, пошук та фільтрацію даних:
    1) сортувати прилади за номером замовлення, датою проведення повірки;
    2) здійснювати пошук інформації про прилад за його повним або частковим заводським номером;
    3) здійснювати фільтрацію інформації по статусу проведення повірки, по факту оплати рахунку фактури;
- система повинна підтримувати можливість додавання, редагування та видалення даних про замовників, виконавців, замовлення, прилади, тощо;
- система повинна підтримувати формування необхідних статистик типу статистики щодо виконаних замовлень, щодо навантаження співробітників, повірки приладів по категоріям, географії замовників і т.п.;
- система повинна підтримувати підготовку та друк наступних звітів, наприклад, рахунок-фактури на проведення повірки, звіт з навантаження співробітників, тощо.

Таким чином, для розробки програмної системи необхідно виконати наступні задачі:

- провести аналіз та моделювання предметної області обліку метрологічних приладів;
- спроектувати базу даних для збереження інформації з предметної області;
- розробити алгоритми підтримки бізнес-процесу метрологічного обліку приладів;
- спроектувати архітектуру програмної системи;
- виконати програмну реалізацію серверної та клієнтської частин системи та провести тестування створеної програмної системи.

1.3.1 Цільова аудиторія

Основною цільовою аудиторією програмної системи будуть:

- співробітники лабораторій та підрозділів установ типу Облстандартметрології;
- начальники відповідних лабораторій та підрозділів;
- після розвитку системи з додаванням веб-частини також замовники, що є представниками підприємств та організацій, що використовують метрологічні прилади.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Вимоги до програмної системи є сукупністю показників якості та функцій, які повинна підтримувати програмна система. Визначення таких вимог є важливою частиною життєвого циклу програмної системи, адже від правильності визначених цих вимог залежить в подальшому простота розробки.

Всі вимоги до програмних систем діляться на функціональні та не функціональні. Функціональні вимоги описують поведінку програмної системи, не функціональні вимоги стосуються параметрів побудови системи, її архітектури, безпеки, тощо.

В роботі необхідно створити програмну систему метрологічного обліку приладів для систем типа Облстандартметрологія. Система повинна складатися з клієнтської частини, яка надає користувачу дружній інтерфейс та забезпечує основний функціонал, та серверної частини, яка є сервером БД, що забезпечує роботи з об’єктами БД.

2.1 Функціональні вимоги

Програмна система розробляється для підтримки метрологічного обліку законодавчо регульованих засобів вимірювальної техніки на основних етапах цього обліку, від прийому заявки від замовника до виписки сертифікату на результати повірки. Програмний продукт має надавати зручні та ефективні можливості для прийому заявок від замовників, роботи співробітників відділів та лабораторій на різних етапах обслуговування заявки.

Програмна система складається з клієнтської та серверної частин. Інтерфейс системи буде реалізований українською мовою, оскільки система направлена на використання в державному секторі. В системі повинні бути передбачено функціонал для адміністрування бази даних.

Програмна система повинна забезпечуватимуть наступну функціональність:

- авторизація користувачів;
- реєстрація заявок на повірку приладів;
- зміна стану заявки під час проходження основних етапів її обробки;
- отримання статистик щодо роботи відділів, обробки замовлень, роботи співробітників, тощо;
- формування документів, що супроводжують життєвий цикл заявки;
- керування даними системи, а саме, додавання, редагування та видалення даних з довідкових таблиць системи.

Програмна система повинна надавати функціонал для двох типів користувачів:

- адміністратор системи;
- співробітник метрологічної організації.

Адміністратор системи буде виконувати роботу з акаунтами користувачів та довідковими таблицями системи.

Співробітник підприємства працює з усім інших основним функціоналом системи.

2.2 Не функціональні вимоги

До не функціональних вимог до програмної системи відносяться:

- доступність – програмна система повинна бути доступною для користувачів в разі, коли є з’єднання з БД;
- швидкодія – відповідь на запити до БД повинні бути швидшими за 2 секунди;
- надійність – програмна система повинна працювати надійно, в разі виникнення помилок користувачі повинні отримувати повідомлення.

Серверна частина системи повинна реалізовувати певну частину бізнес-логіку, що буде зменшувати кількість обчислень на клієнтській стороні, що, фактично, зменшить системні вимоги до клієнтської частини.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз та моделювання предметної області

Для забезпечення єдності вимірів метрологічними підрозділами України проводиться повірка та калібрування вимірювальних приладів у багатьох сферах науки та виробництва.

Згідно наказу №193 від 08.02.2016 року Міністерства економічного розвитку і торгівлі України «Про затвердження порядку проведення повірки законодавчо регульованих засобів вимірювальної техніки, що перебувають в експлуатації, та оформлення її результатів» [1], постанови КМУ №374 від 04.06.2015 року [9] «Про затвердження переліку категорій законодавчо регульованих засобів вимірювальної техніки (ЗР ЗВТ), що підлягають періодичній повірці» та, відповідно до статті 17 Закону України «Про метрологію та метрологічну діяльність» від 05.06.2014 р № 1314-VII всі ЗР ЗВТ [10], що знаходяться в експлуатації, підлягають періодичній повірці і повірці після ремонту.

Повірку ЗР ЗВТ можуть виконувати сертифіковані лабораторії та підрозділи Облстандартметрології, які розташовані в багатьох обласних центрах нашої країни.

Структура метрологічних підприємств включає в себе відділи, кожен з яких обладнаний щодо перевірки певної групи приладів. Відділом керує начальник відділу, повірку та оформлення результатів забезпечують безпосередньо співробітники відділу, сертифіковані інженери-метрологи. Замовниками послуг є співробітники метрологічної служби підприємств або ФОП, діяльність яких відноситься до переліку категорій ЗР ЗВТ.

Мета роботи відділу метрологічного підприємства – організація та підтримка процесу повірки ЗР ЗВТ, до якого входять замовники, виконавці та прилади, яким треба провести повірку.

Далі будемо розглядати процес проведення повірки ЗР ЗВТ на прикладі роботи відділу теплотехнічних вимірювань.

Замовники звертаються безпосередньо в відділ з листом, в якому просять провести повірку їх приладів. Співробітник відділу реєструє замовника та лист в системі, після цього автоматично формується замовлення та рахунок на оплату послуг.

Коли замовник здійснює оплату рахунку та надає прилади до відділу, на протязі 15 робочих днів повинна бути виконана повірка приладів та оформлені свідоцтва про повірку.

Інформаційні потреби співробітників відділу:

- необхідна повна інформація про підприємство замовника, контактну особу замовника, кількість та тип приладів для повірки;
- необхідно мати дані про технічні параметри різних типів приладів (діапазон вимірювання, точність, допустима похибка, тощо), вартість повірки;
- легко визначати поточний стан оплати замовлення та повірки конкретного приладу;
- співробітники повинні мати змогу формувати звіти з результатами проведення повірок за місяць.

Основні потреби користувачів системи можна представити за допомогою UML-діаграми - USE-CASE (див. рис. 3).

Як бачимо, користувач, який є виконавцем замовлень в організації може авторізуватися в системі, вести облік замовників та обробляти замовлення. Обробка замовлень включає в себе формування рахунок-фактури, повірку приладів та виписку сертифікатів на прилади.

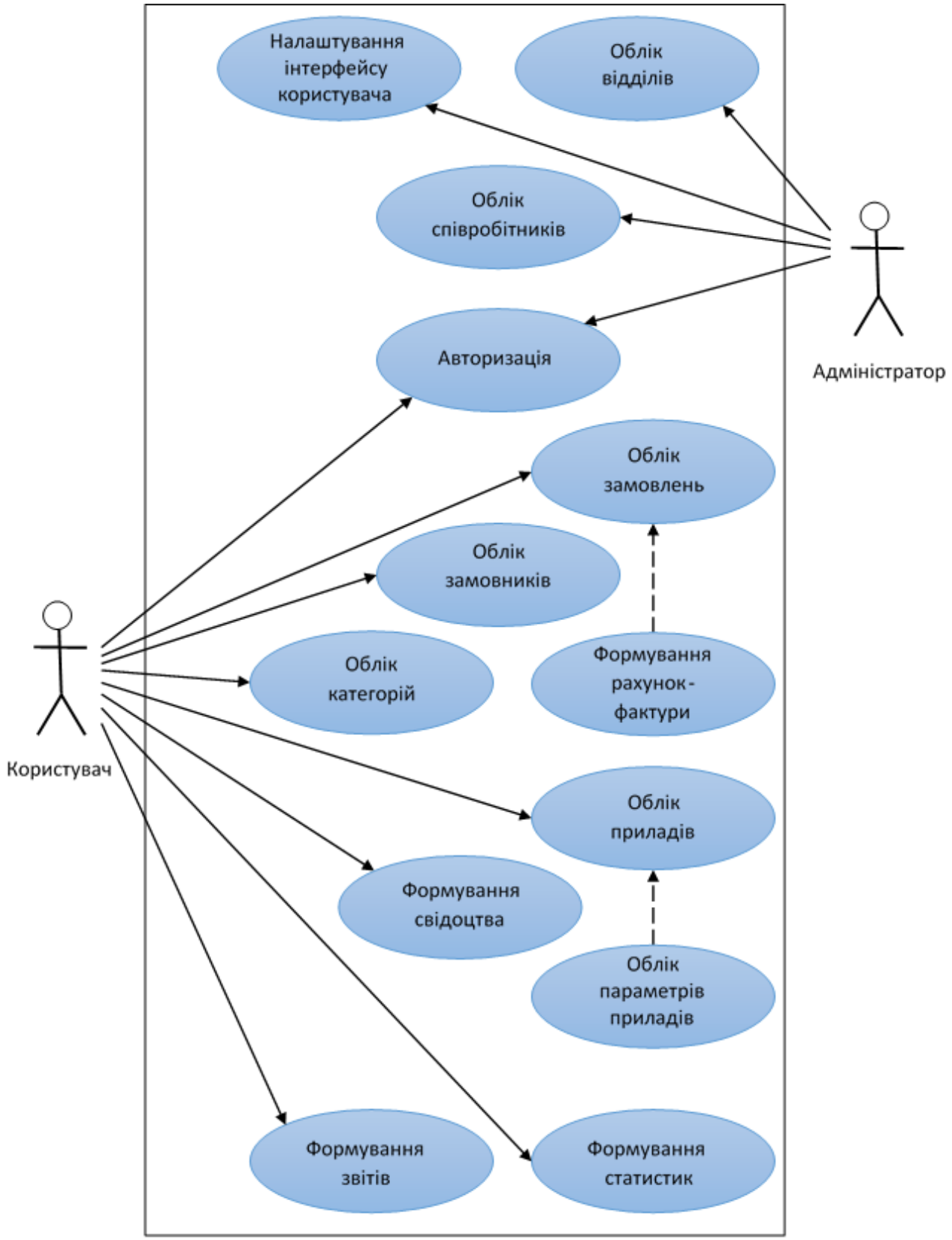


Рисунок 3 – Загальна USE-CASE діаграма (рисунок виконано самостійно)

Опис об’єктів предметної області та зв'язків між ними зобразимо у вигляді загальної діаграми класів (див. рис. 4).

Зазначимо перелік характеристик для кожного з наведених понять. Поняття замовник має наступні властивості: назва підприємства, його адреса, код ЕДРПОУ, електронна пошта підприємства, ПІБ контактної особи (КО), електронна пошта КО, телефон КО. Поняття виконавець включає в себе ПІБ виконавця, його посада, телефон. Прилади мають назву, заводський номер, діапазон вимірювання, припустиму похибку.

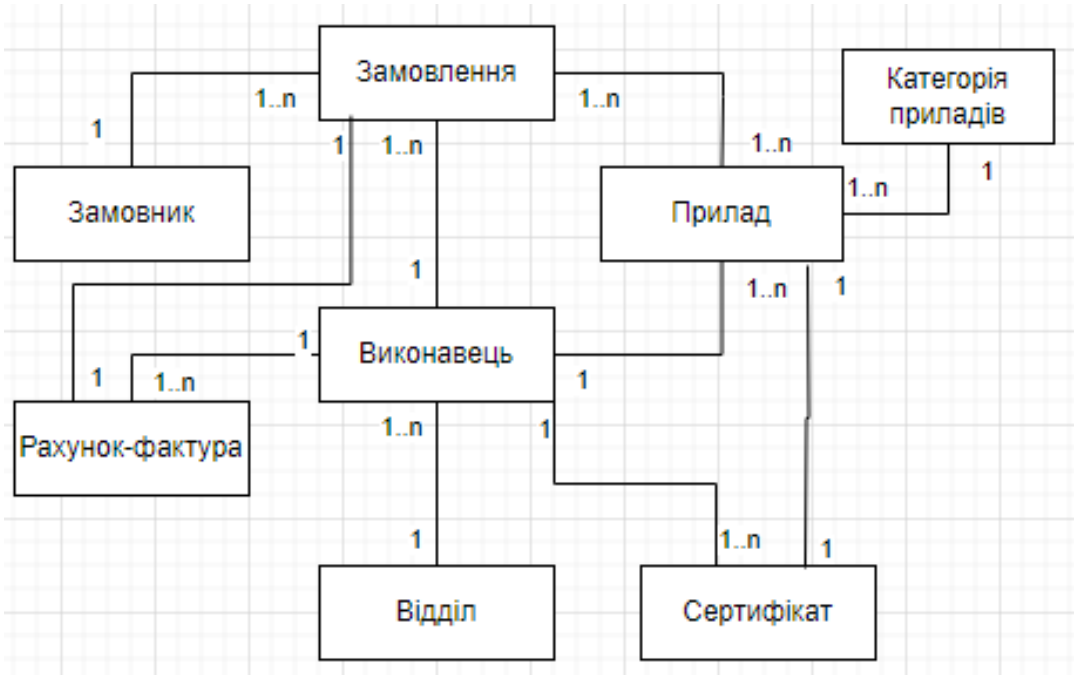


Рисунок 4 – Загальна діаграма класів (рисунок виконано самостійно)

Для спрощення роботи виконавців під час роботи з інформацією вони повинні мати можливість:

- сортувати прилади по даті повірки, по номеру замовлення, по ЕДРПОУ замовника;
- здійснювати пошук інформації про прилад за його повним або частковим заводським номером;
- здійснювати фільтрацію інформації по статусу проведення повірки, по факту оплати рахунку фактури;
- додавати та редагувати дані про замовників та прилади.

Співробітники відділу для спрощення аналізу інформації з процесу проведення повірки приладів повинні мати можливість отримання наступної статистики:

- статистику виконаних замовлень (період, кількість виконаних замовлень, загальна сума);
- статистику навантаження співробітників (ПІБ співробітника, період, кількість виконаних заявок, загальна сума);
- статистику повірки приладів по категоріям (період, категорія приладів, кількість приладів по категоріям, які пройшли повірку, які не пройшли повірку);
- статистику географії замовників (період, місто, кількість замовників);

Опис існуючого документообігу складається з наступних документів:

- рахунок-фактура на проведення повірки (замовник, номер замовлення, перелік категорій приладів, кількість приладів кожної категорії, вартість повірки, ПДВ, загальна вартість, ПІБ виконавця(підпис), ПІБ начальника відділу(підпис));
- звіт з навантаження співробітників (ПІБ співробітника, місяць, кількість виконаних заявок, загальна сума);

Описання алгоритмічних залежностей показників в ПО роботи системи метрологічного обліку приладів:

- строк дії свідоцтва = дата оформлення свідоцтва + міжповірочний інтервал типу приладу;
- дата закінчення повірки = дата початку повірки + 21 день;
- загальна вартість = вартість + ПДВ

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно зв’язків:

- кожен замовник може надати для повірки один чи багато приладів;
- виконавець може одночасно повіряти один чи багато приладів;

– виконавець може обробляти замовлення багатьох замовників, а замовники можуть відправляти багато замовлень до виконавців.

3.2 Проектування бази даних

Під час проектування бази даних на основі проведеного аналізу була розроблена інфологічна модель (ER-діаграма) БД за нотацією Баркера, що наведена на рисунку 5.

Для розробки БД була обрана реляційна модель. Реляційна база даних – це база даних, яка ґрунтуються на інтуїтивно зрозумілому, наочному поданні інформації у вигляді таблиць. Кожен рядок у таблиці такої бази даних являє собою запис унікальним ідентифікатором – ключем. Стовпці таблиць мають атрибути даних, що дає змогу встановлювати зв’язки між елементами даних. Схема бази даних системи приведена на рисунку 6.

Розроблена БД була перевірена на відповідність вимогам нормальних форма. Нормальна форма – це певна вимога-обмеженн, що висувається до структури таблиць в теорії реляційних БД для усунення з бази зайвих функціональних залежностей між атрибутами. Якщо у відношенні всі атрибути є атомарними, тоді воно знаходяться в 1 НФ. При приведенні до другої нормальної форми було доведено, що всі відношення не містять неповних функціональних залежностей, тобто в складі потенційного ключа відсутня менша підмножина атрибутів, від якої можна також вивести дану функціональну залежність. При приведенні до третьої нормальної форми була проведена перевірка на відсутність транзитивних функціональних залежностей.

Таким чином, розроблена схема БД знаходиться в 3НФ і на її основі може бути створена фізична модель БД.

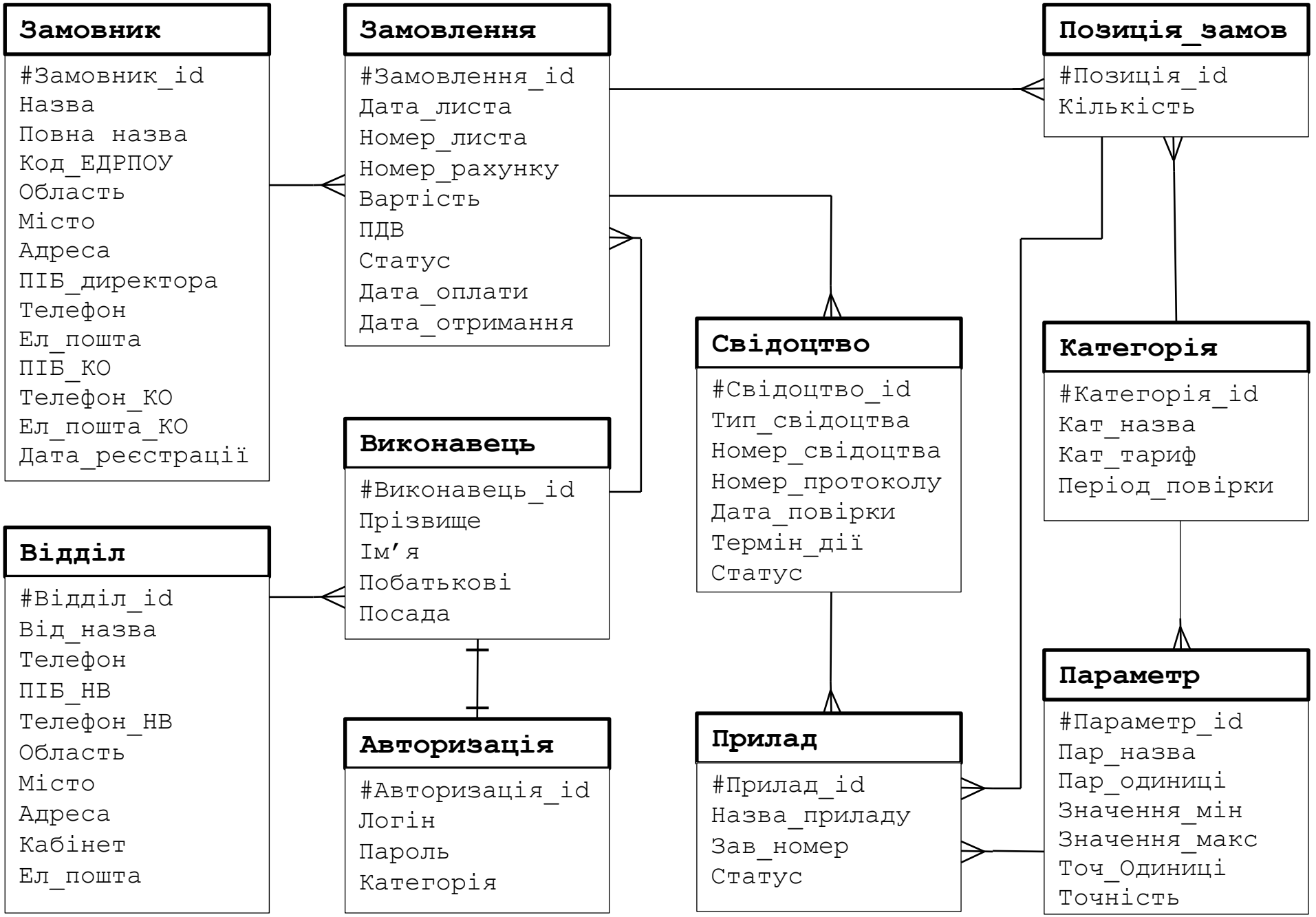


Рисунок 5 – ER-діаграма бази даних (рисунок виконано самостійно)

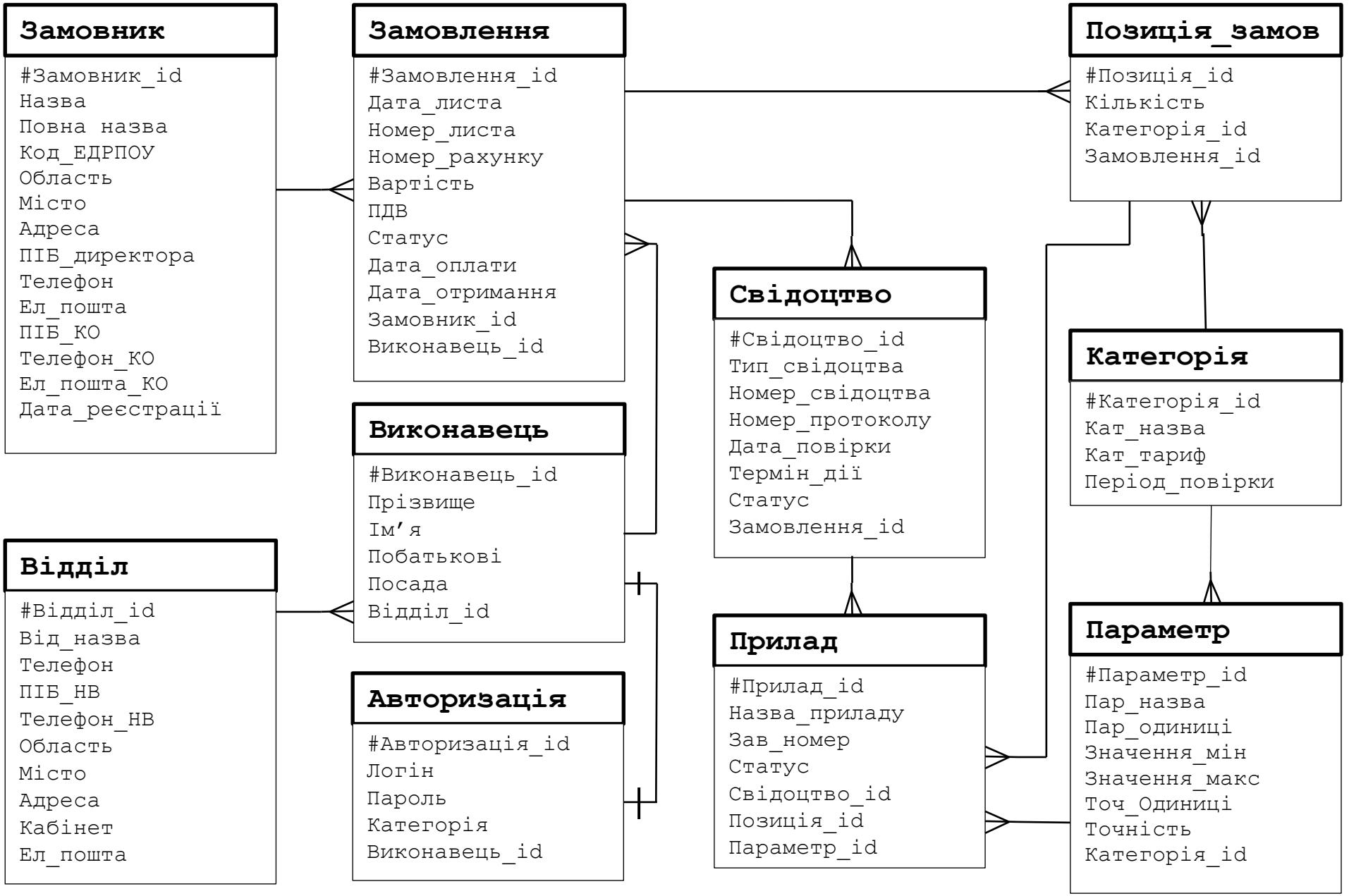


Рисунок 6 – Схема бази даних (рисунок виконано самостійно)

3.3 Розробка алгоритмів підтримки бізнес-процесу обліку приладів

На основі аналізу предметної області було виділено основний бізнес-процес, який повинна підтримувати програмна система – обробка замовлень на повірку МП. В даному бізнес процесі приймають участь замовники та співробітники підрозділів та лабораторій метрологічної організації (виконавці).

На рисунку 7 наведено діаграму потоків даних, що відображає основні процеси цієї бізнес-задачі та структури даних, які будуть використовуватися для їх підтримки.

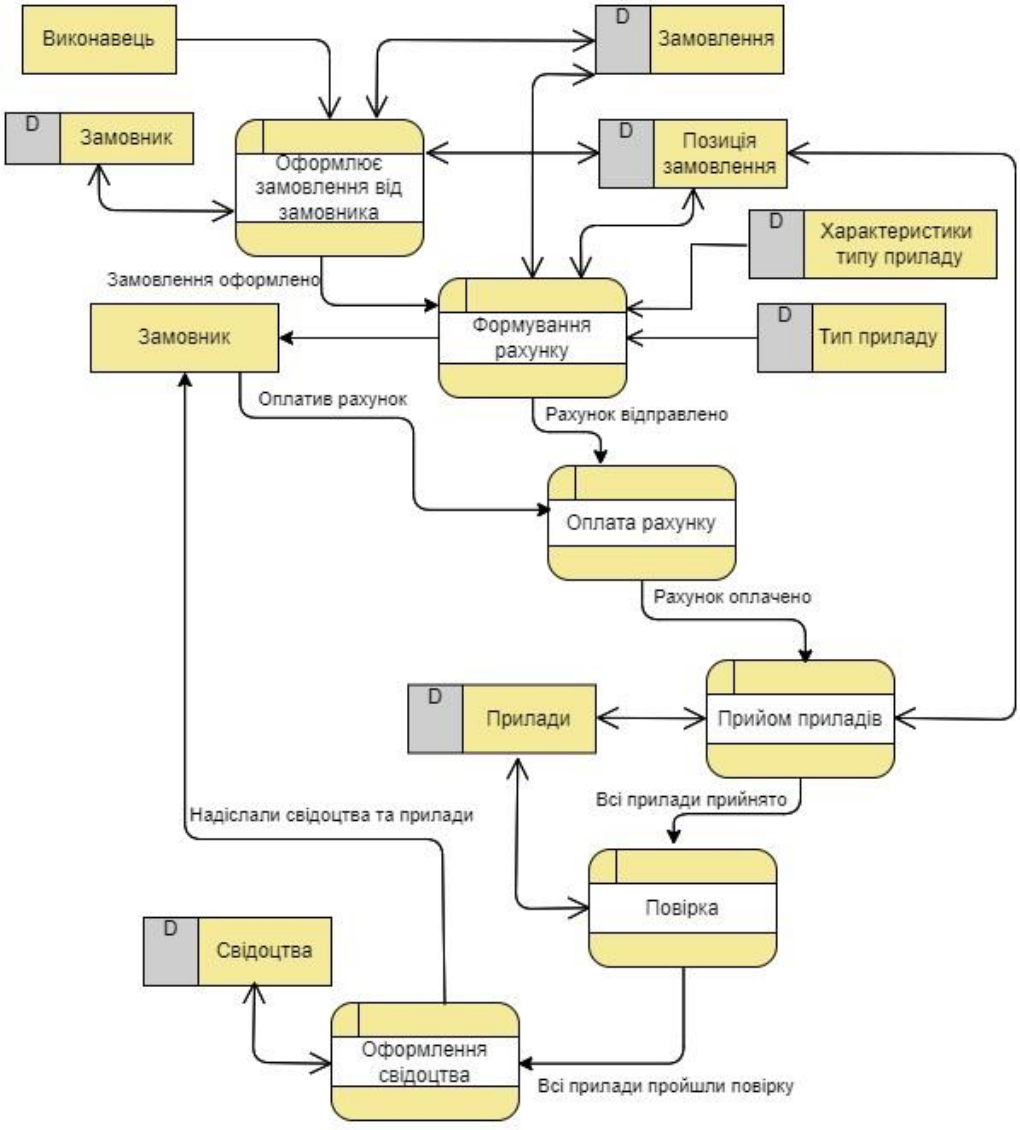


Рисунок 7 – Діаграма потоків даних (DFD) для процесу обробки замовлення (рисунок виконано самостійно)

Виходячи з DFD діаграми було виявлено реактивні об’єкти, поведінка яких найкраще характеризується його реакцією на події, одержувані ним ззовні його контексту. Реактивний об'єкт має чіткий час життя, і його поточна поведінка залежить від минулого. Зазвичай реактивний об'єкт простоює в очікуванні події. Коли він отримує цю подію, його реакція зазвичай залежить від попередніх подій. Відреагувавши певним чином, об'єкт знову повертається в стан простою, чекаючи наступної події.

В якості таких об’єктів в предметній області були вибрані Замовлення та Прилад. На рисунку 8 наведено діаграму станів для об’єкту Замовлення. На рисунку 9 наведено діаграму станів для об’єкту Прилад.

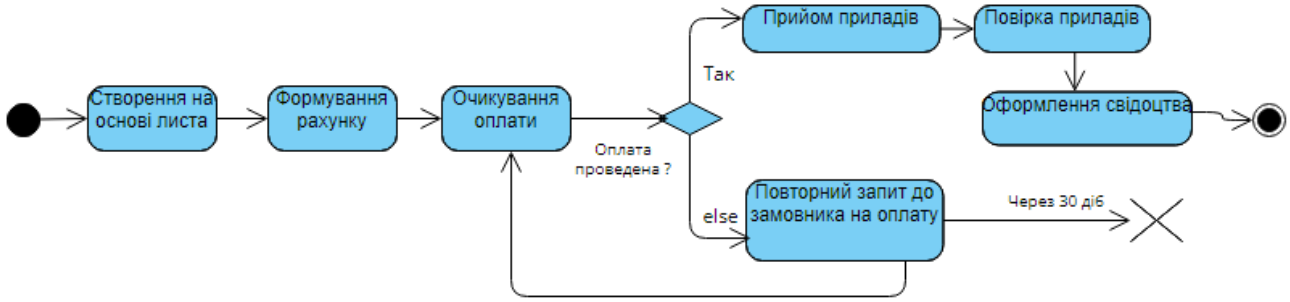


Рисунок 8 – Діаграма станів замовлення (рисунок виконано самостійно)

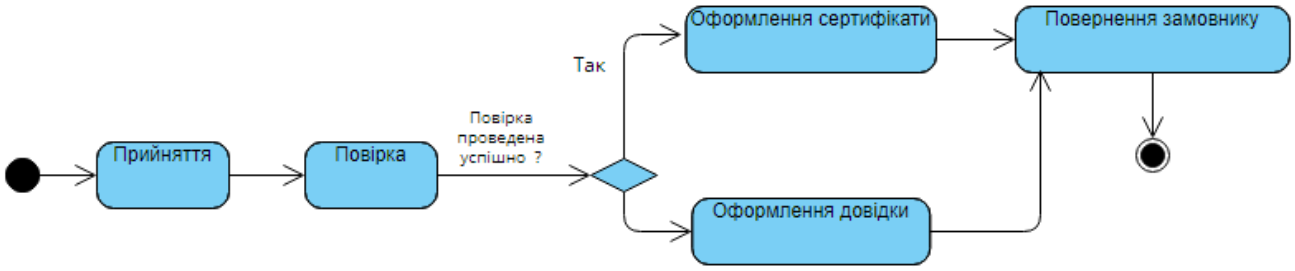


Рисунок 9 – Діаграма станів приладу (рисунок виконано самостійно)

Розроблені діаграми станів дозволять в подальшому реалізувати відповідну логіку для керування станами розглянутих об’єктів, яку можна буде реалізувати на серверній стороні системи. А DFD дозволить організувати логіку на клієнтській стороні системи.

3.4 Розробка архітектури системи

В якості моделі архітектури для створюваної програмної системи був обраний шаблон клієнт-сервер з бізнес-логікою на сервері. В цій архітектурі сервер розміщує та керує більшістю ресурсів, що споживає клієнт. Такий тип архітектури має один або декілька клієнтських комп'ютерів, підключених до сервера бази даних через інтернет-з'єднання.

В цьому випадку використовується можливість сучасних серверів БД виконувати збережені SQL процедури на сервері, куди і переноситься максимально можлива частина бізнес-логіки. Вимоги до сервера БД зростають, однак різко знижуються вимоги до клієнтських машин (за рахунок виносу з них бізнес-логіки) і до пропускної здатності мережі (клієнту передаються тільки дані, необхідні користувачеві) (див. рис. 10).

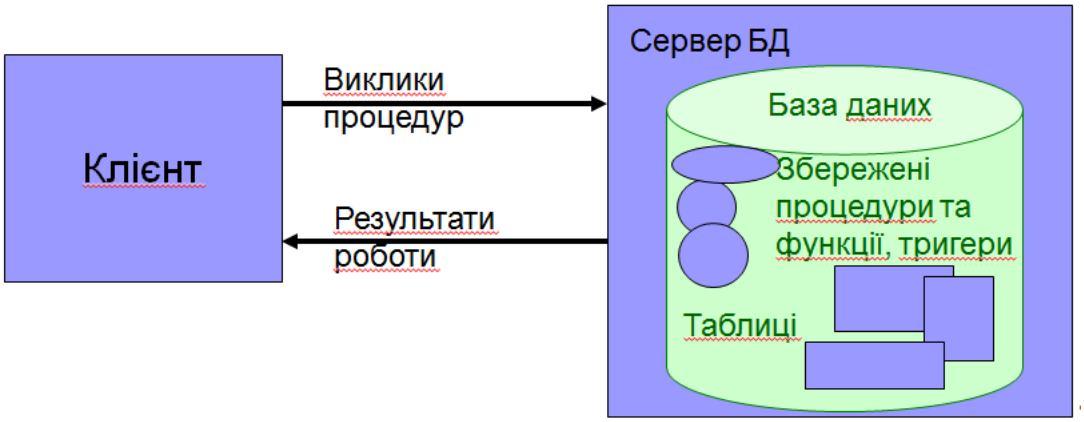


Рисунок 10 – Загальне надання архітектури клієнт-сервер з бізнес-логікою на сервері (рисунок виконано самостійно)

До переваг такої архітектури можна віднести:

– достатньо низькі вимоги до пропускної здатності мережі;
– простіший процес створення бізнес-логіки;
– дозволяє тримати велике навантаження.

До недоліків такої архітектури можна віднести:

– підвищені вимоги до сервера БД;

– невисоку переносимість (мобільність) системи на інші сервери БД.

Для доступу к даним та взаємодії клієнта та сервера буде використовуватися технологія ADO.NET [11]. В ADO.NET використовується модель роботи користувача у відриві від джерела даних. Програма взаємодіє з базою даних тільки на невеликий проміжок часу. З'єднання встановлюється тільки тоді, коли клієнт з віддаленого комп'ютера запитує на сервері дані. Після того, як сервер підготував необхідний набір даних, сформував і відправив їх клієнту, зв'язок додатка з сервером відразу ж обривається, і клієнт переглядає отриману інформацію вже не в зв'язку з сервером.

ADO.NET орієнтується на три важливі можливості:

– підтримка моделі доступу до непов'язаних даними;
– підтримка тісної інтеграції з XML;
– інтеграція з .NET Framework, а саме сумісність з базовою бібліотекою класів типової системи.

В ADO.NET як єдиний формат для передачі даних використовується XML-формат. Тобто, якщо дані записати в якийсь зовнішній файл, то вони будуть записані в XML-форматі. Таким чином, файл з даними у форматі XML є точно таким же джерелом, як і будь-яка база даних, і з нього можна помістити дані в об'єкт DataSet на клієнті. Фактично в ADO.NET XML є фундаментальним форматом для даних. У ADO.NET автоматично створюють XML-файли при передачі інформації між об'єктами і компонентами.

Використання обміну даних в єдиному, форматі XML має такі переваги:

– XML - єдиний промисловий стандарт, що дозволяє розробленим компонентів обмінюватися даними з будь-якими іншими компонентами в будь-яких інших додатках, оскільки всі вони розуміють і використовують формат XML;
– при поданні даних в XML не використовується двійкове кодування, а це забезпечує можливість передачі даних через будь-який протокол, наприклад, через HTTP;

– ADO.NET автоматично виконає перетворення даних в XML і з XML-формату так, як потрібно в поточний момент; розробник же буде взаємодіяти з даними шляхом використання властивостей та методів об'єктів.

Для опису архітектури програмної системи, що створюється, була розроблена діаграма розгортання, яка описує фізичне розгортання інформації, яка генерується системою на апаратних компонентах. Діаграма представлена на рисунку 11.

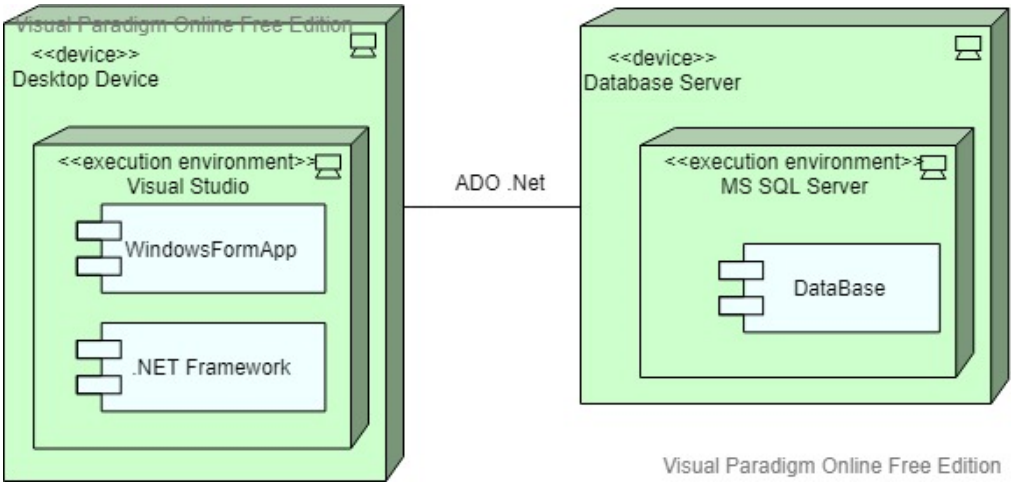


Рисунок 11 - Діаграма розгортання програмної системи (рисунок виконано самостійно)

Основною частиною клієнт-серверною архітектури є сервер, тому потрібно належним чином обрати СКБД, на базі якої буде створюватися сервер. Необхідно, щоб окрім мови SQL СКБД підтримувала процедурне розширення. Отже, було обрано СКБД MS SQL Server, яка має процедурну мову Transact-SQL, що дозволяє створювати тригери та процедури і функції для реалізації бізнес-логіки серверної частини.

Для доступу до БД з клієнтської частини використовується рядок з’єднання з базою даних. Вхід до бази даних відбувається за допомогою авторизації. В якості постачальника даних за технологією ADO.NET обрано SQL

Server .NET Data Provider, що оптимізований для роботи з Microsoft SQL Server V7.0 і вище.

Усі запити на сервер мають бути захищеними та відправлятися протоколом HTTPS.

3.5 Створення UI / UX або іншого дизайну системи

Процеси проектування (UI) користувацького інтерфейсу та (UX) користувацького досвіду є ключовими моментами в розробці будь-якої програмної системи. Для програмної системи метрологічного обліку приладів було розроблено навігаційну схему системи (див. рис. 12), на якій відображено основні можливості користувачів та адміністратору системи.

Основна структура навігації буде включати наступні розділи:

- вікно авторизації: центральна точка входу в систему, на якій користувач повинен ввести реєстраційну інформацію і відповідно отримати доступ до основних функцій системи;
- форма обліку замовлень: містить основний функціонал, що дозволяє прийняти та відредагувати стан замовлення на повірку приладів;
- форма Адміністрування дозволяє користувачу з роллю «Адміністратор» керувати користувачами: додавати їх, редагувати їх дані, в тому числі, паролі, та видаляти їх ;
- Замовники: форма для керування інформацію про замовників;
- Виконавці: форма для керування інформацією про співробітників лабораторій та відділів організації, що виконують повірку метрологічних приладів;
- Прилади: сторінка для керування приладами, що проходять метрологічний огляд;
- Категорії приладів: сторінка довідника з категоріями приладів;
- Сертифікати: сторінка роботи з сертифікатами, що отримано на прилади, які було повірено;
- тощо.

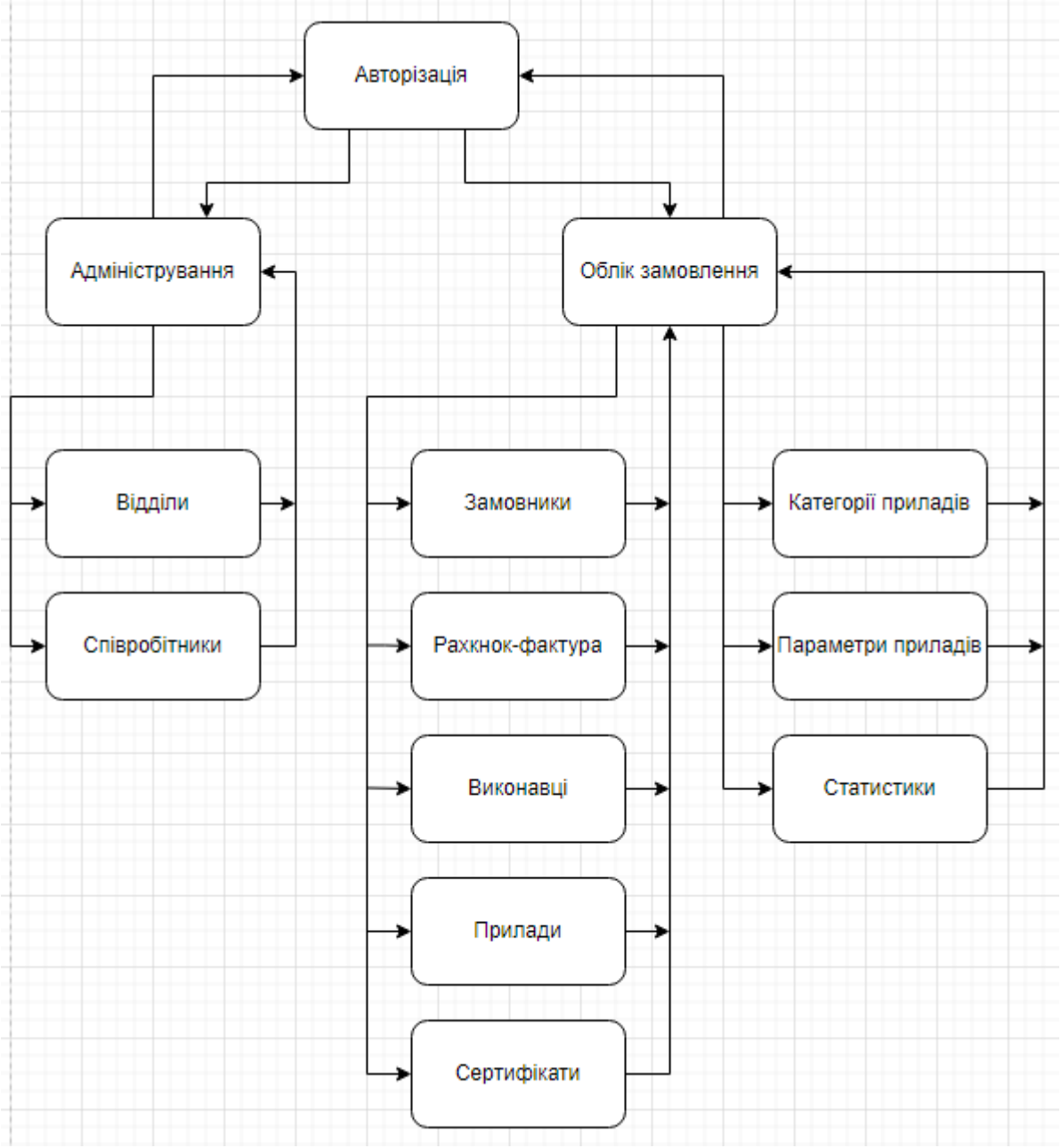


Рисунок 12 – Схема навігації по системі обліку метрологічного приладів (рисунок виконаний самостійно)

Для адміністрування створена окрема форма, на яку можна окремо потрапити в разі авторизації як Admin та отримати можливості управління даними співробітників організації.

4 ОПИС ПРОГРАМИ

4.1 Вибір засобів розробки

Для втілення даної програмної було використано середу розробки Microsoft Visual Studio 2022 та мову програмування C#. Для створення візуального представлення програми було обрано інтерфейс програмування додатків Windows Forms, що є частиною Microsoft .NET Framework [12]. Даний інтерфейс дозволяє швидко створити зручний та інтуїтивно зрозумілий додаток.

Додаток націлено на взаємодію із базами даних, отже для їх створення було обрано систему керування реляційними базами даних Microsoft SQL Server [13]. Саме ця СКБД була використана через її здатність швидкої обробки інформації, легкість використання та інтегрованість з іншими продуктами Microsoft. За допомогою утіліти SQL Server Management Studio було здійснено управління та конфігурування компонентів Microsoft SQL Server.

Доступ до баз даних було здійснено за технологією ADO.NET [11]. Одним з головних плюсів цієї технології є використання роз’єднаної моделі доступу до даних. Раніше при створенні застосунків традиційно використовувалась технологія доступу до джерела даних за якої з'єднання з базою підтримується постійно. Хоча постійне з'єднання дає змогу дещо прискорити роботу застосунку, загальний збиток від розтрати системних ресурсів зводить перевагу нанівець. Специфіка розподілених систем не дає змоги серверу в кожен момент часу знати, що необхідно користувачеві. Тобто до наступного запиту сервер не має уявлення, чи потрібно ще підтримувати з'єднання. Саме тому було надано перевагу ADO.NET, де з'єднання встановлюється лише на той короткий час, коли необхідно проводити операції над базою даних. Але ж ця технологія не буде доречною при використанні програм, які проводять часті та об'ємні зміни змісту записів.

4.2 Призначення і логічна структура

Програмна система «Метрологічний облік приладів» призначена для співробітників підрозділів Облстандартметрології. Вона дозволяє автоматизувати роботу по обліку замовників, замовлень та приладів при проведенні повірки законодавчо регульованих засобів вимірювальної техніки.

Логічна структура додатку приведена на рисунку 13.

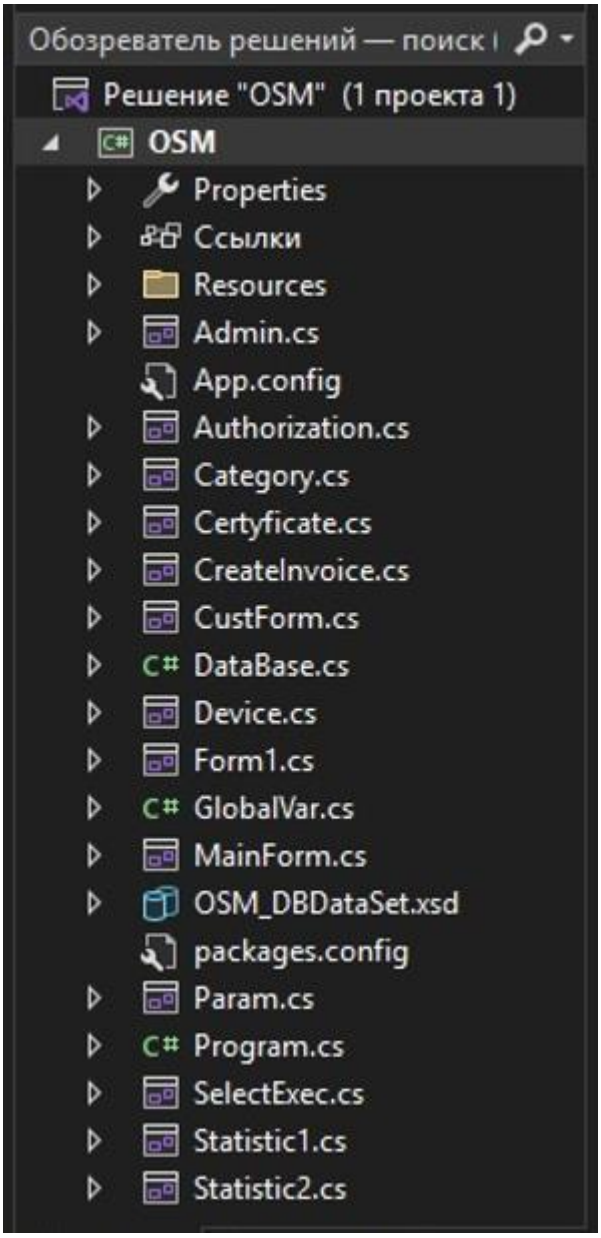


Рисунок 13 – Логічна структура системи «Метрологічний облік приладів» (рисунок виконано самостійно)

Опис форми «Адміністрування» та «Налаштування» та алгоритми їх роботи приведено в класах AdminForm, DepForm та SetForm. Головне вікно програми описано в класі MainForm.

Робота з категоріями приладів та приладами відбувається за допомогою класів Category та Device. Робота з замовниками відбувається за допомогою класу CustForm, а оформлення сертифікатів відбувається за допомогою класу Certyficate.

Обробка статистик – в класах Statistic1 та Statistic2.

4.3 Описання фізичної моделі бази даних

При реалізації фізичної моделі бази даних було створено десять таблиць. Нижче наведено команди на їх створення.

```
USE [OSM_DB]
GO

CREATE TABLE [dbo].[Authorizations](
[Auth_id] [int] IDENTITY(1,1) NOT NULL,
[Auth_login] [varchar](30) NULL,
[Auth_password] [varchar](20) NULL,
[Auth_category] [int] NULL,
[Ex_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
[Auth_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=    OFF,    ALLOW_ROW_LOCKS    =    ON,    ALLOW_PAGE_LOCKS    =    ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Category](
[Cat_id] [int] IDENTITY(1,1) NOT NULL,
[Cat_name] [varchar](50) NULL,
[Cat_tariff] [float] NULL,
[Verification_period] [int] NULL,
PRIMARY KEY CLUSTERED
(
[Cat_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=    OFF,    ALLOW_ROW_LOCKS    =    ON,    ALLOW_PAGE_LOCKS    =    ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Certificat](
[Cer_id] [int] IDENTITY(1,1) NOT NULL,
[Cer_type] [varchar](20) NULL,
```

```
	[Cer_num] [int] NULL,
	[Protokol_num] [int] NULL,
	[Verification_date] [date] NULL,
	[Expiration_date] [date] NULL,
	[Ord_id] [int] NULL,
	PRIMARY KEY CLUSTERED
	(
	[Cer_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=	OFF,	ALLOW_ROW_LOCKS	=	ON,	ALLOW_PAGE_LOCKS	=	ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
	) ON [PRIMARY]


	CREATE TABLE [dbo].[Customer](
	[Cust_id] [int] IDENTITY(1,1) NOT NULL,
	[Cust_full_name] [varchar](150) NULL,
	[Cust_name] [varchar](50) NULL,
	[Cust_EDRPOU] [int] NULL,
	[Cust_region] [varchar](20) NULL,
	[Cust_city] [varchar](20) NULL,
	[Cust_address] [varchar](50) NULL,
	[Cust_head_name] [varchar](50) NULL,
	[Cust_tel] [varchar](13) NULL,
	[Cust_email] [varchar](30) NULL,
	[Cust_KO_name] [varchar](50) NULL,
	[Cust_KO_tel] [varchar](13) NULL,
	[Cust_KO_email] [varchar](30) NULL,
	[Cust_date_reg] [date] NULL,
	PRIMARY KEY CLUSTERED
	(
	[Cust_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=	OFF,	ALLOW_ROW_LOCKS	=	ON,	ALLOW_PAGE_LOCKS	=	ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
	) ON [PRIMARY]


	CREATE TABLE [dbo].[Department](
	[Dep_id] [int] IDENTITY(1,1) NOT NULL,
	[Dep_name] [varchar](50) NULL,
	[Dep_tel] [varchar](13) NULL,
	[Dep_nameHD] [varchar](50) NULL,
	[Dep_HD_tel] [varchar](13) NULL,
	[Dep_region] [varchar](20) NULL,
	[Dep_city] [varchar](20) NULL,
	[Dep_address] [varchar](50) NULL,
	[Dep_office] [int] NULL,
	[Dep_email] [varchar](30) NULL,
	[Dep_nums] [int] NULL,
	PRIMARY KEY CLUSTERED
	(
	[Dep_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=	OFF,	ALLOW_ROW_LOCKS	=	ON,	ALLOW_PAGE_LOCKS	=	ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
	) ON [PRIMARY]
```

```
CREATE TABLE [dbo].[Device](
[Dev_id] [int] IDENTITY(1,1) NOT NULL,
[Dev_name] [varchar](50) NULL,
[Dev_num] [varchar](20) NULL,
[Dev_status] [varchar](20) NULL,
[Cer_id] [int] NULL,
[Pos_id] [int] NULL,
[Par_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
[Dev_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=    OFF,    ALLOW_ROW_LOCKS    =    ON,    ALLOW_PAGE_LOCKS    =    ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Executor](
[Ex_id] [int] IDENTITY(1,1) NOT NULL,
[Ex_surname] [varchar](50) NULL,
[Ex_name] [varchar](50) NULL,
[Ex_patname] [varchar](50) NULL,
[Ex_position] [varchar](50) NULL,
[Dep_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
[Ex_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=    OFF,    ALLOW_ROW_LOCKS    =    ON,    ALLOW_PAGE_LOCKS    =    ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [dbo].[Orders](
[Ord_id] [int] IDENTITY(1,1) NOT NULL,
[Ord_let_num] [varchar](10) NULL,
[Ord_let_date] [date] NULL,
[Ord_inv_num] [int] NULL,
[Ord_cost] [float] NULL,
[Ord_VAT] [float] NULL,
[Ord_pay_date] [date] NULL,
[Ord_rec_date] [date] NULL,
[Cust_id] [int] NULL,
[Ex_id] [int] NULL,
[Ord_status] [varchar](20) NULL,
PRIMARY KEY CLUSTERED
(
[Ord_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=    OFF,    ALLOW_ROW_LOCKS    =    ON,    ALLOW_PAGE_LOCKS    =    ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [dbo].[Parameter](
[Par_id] [int] IDENTITY(1,1) NOT NULL,
[Par_name] [varchar](30) NULL,
[Par_units] [varchar](10) NULL,
[Min_value] [float] NULL,
```

```
	[Max_value] [float] NULL,
	[Accuracy_units] [varchar](10) NULL,
	[Accuracy_value] [float] NULL,
	[Cat_id] [int] NULL,
	PRIMARY KEY CLUSTERED
	(
	[Par_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=	OFF,	ALLOW_ROW_LOCKS	=	ON,	ALLOW_PAGE_LOCKS	=	ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
	) ON [PRIMARY]

	CREATE TABLE [dbo].[Position](
	[Pos_id] [int] IDENTITY(1,1) NOT NULL,
	[Pos_quantity] [int] NULL,
	[Cat_id] [int] NULL,
	[Ord_id] [int] NULL,
	PRIMARY KEY CLUSTERED
	(
	[Pos_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
=	OFF,	ALLOW_ROW_LOCKS	=	ON,	ALLOW_PAGE_LOCKS	=	ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
	) ON [PRIMARY]
```

Формування та опис зовнішніх ключів після виконання всіх команд для побудови таблиць дозволив описувати сутності бази даних у довільному порядку.

4.4 Опис програмної реалізації серверної частини системи

Відповідно до діаграми станів замовлення, що було розроблено в розділі 3.3 (див. рис. 8), було розроблено тригери, які реалізують логіку переходу замовлення з одного стану до іншого.

По-перше, від час створення замовлення, воно може прийняти статус «Лист отримано». На серверній частині створено тригер, який унеможливить додавання замовлення з іншим статусом.

```
	CREATE OR ALTER TRIGGER trg_insert_order
	ON orders
	AFTER INSERT AS
	BEGIN
		UPDATE orders
		SET ord_status = 'Лист отримано'
		FROM orders o
		JOIN inserted i ON o.ord_id = i.ord_id;
	END;
```

Так само, від час створення запису про новий прилад, прилад може прийняти лише статус «Отримано». На серверній частині створено тригер, який унеможливить додавання приладу з іншим статусом.

```
CREATE OR ALTER TRIGGER trg_insert_device
ON device
AFTER INSERT AS
BEGIN
        UPDATE device
        SET dev_status = 'Отримано'
        FROM device dev
        JOIN inserted i ON dev.dev_id = i.dev_id;
END;
```

Також перехід до наступного стану замовлення може бути лише з певного попереднього стану. Цю логіку реалізує тригер, що наведено нижче.

```
CREATE TRIGGER trg_update_orders ON orders
AFTER UPDATE AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM inserted i JOIN deleted d ON i.ord_id =
d.ord_id
        WHERE (d.ord_status = 'Лист отримано'
          AND i.ord_status NOT IN ('Рахунок сформовано'))
          OR
             (d.ord_status = 'Рахунок сформовано'
          AND i.ord_status NOT IN ('Рахунок оплачено'))
        OR
         (d.ord_status = 'Рахунок оплачено'
          AND i.ord_status NOT IN ('Прилади отримано'))
        OR
         (d.ord_status = 'Серт. сформовно'
          AND i.ord_status NOT IN ('Виконано'))
    )
    BEGIN
      PRINT 'Помилка! Невірний статус замовлення'
      ROLLBACK TRANSACTION
    END

END;
```

Аналогічний тригер було створено для визначення помилок переходу між станами приладів, який працює відповідно до рисунку 9, що розроблено в розділі 3.3.

Наступний тригер перевіряє, якщо всі прилади, що мають відношення до певного замовлення, повірено, то він переводить замовлення в стан «Прилади повірено».

```
CREATE TRIGGER trg_set_orders_status_believed ON device
AFTER UPDATE AS
DECLARE
     @ord_id int,
     @device_count int,
     @Device_count_in_order int;
 BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN deleted d ON i.dev_id = d.dev_id
        WHERE i.dev_status = 'Повірено'
    )
    BEGIN
        SELECT  @ord_id=p.ord_id
        FROM device dev JOIN deleted d ON dev.dev_id = d.dev_id
        JOIN position p ON dev.pos_id = p.pos_id ;

        SELECT @Device_count=COUNT(dev.Dev_id)
        FROM device dev JOIN position p ON dev.pos_id = p.pos_id
        WHERE dev.dev_status = 'Повірено' AND
              p.ord_id=@ord_id;

        SELECT @Device_count_in_order=SUM(p.pos_quantity)
        FROM position p
        WHERE p.ord_id=@ord_id;

        IF @Device_count_in_order=@Device_count
        BEGIN
          UPDATE orders SET ord_status='Повірка проведена'
          WHERE ord_id=@ord_id;
        END

    END
END;
```

Наступний тригер перевіряє, якщо по всім приладам, що мають відношення до певного замовлення, виписано сертифікати, то він переводить замовлення в стан «Сертифікати сформовано».

```
CREATE TRIGGER trg_orders_status_written ON device
AFTER UPDATE AS
DECLARE
@ord_id int,
@Device_count int,
@Device_count_in_order int;
BEGIN
    IF EXISTS (
        SELECT 1
```

```
        FROM inserted i
        JOIN deleted d ON i.dev_id = d.dev_id
        WHERE i.dev_status = 'Оформлено'
    )
    BEGIN
        SELECT @ord_id=p.ord_id
        FROM device dev JOIN deleted d ON dev.dev_id = d.dev_id
        JOIN position p ON dev.Pos_id = p.Pos_id;

        SELECT @Device_count=COUNT(dev.Dev_id)
        FROM device dev JOIN position p ON dev.Pos_id = p.Pos_id
        WHERE dev.dev_status = 'Оформлено' AND
              p.ord_id=@ord_id;

        SELECT @Device_count_in_order=SUM(p.pos_quantity)
        FROM position p
        WHERE p.ord_id=@ord_id;

        IF @Device_count_in_order=@Device_count
        BEGIN
          UPDATE orders SET ord_status='Серт. сформовано'
          WHERE ord_id=@ord_id;
        END
    END
END;
```

Наведені та інші розроблені тригери дозволили перенести частину логіки з обробки замовлень на серверну частину програмної системи, що дозволило розвантажити клієнтський додаток.

4.5 Опис користувацького інтерфейсу адміністратора

Після запуску програми відкривається головне вікно з заставкою (див. рис. 14), з якого можна перейти до вікна авторизації, зовнішній вигляд якого приведено на рисунку 15. Користувачу треба ввести свій логін і пароль та натиснути кнопку «Увійти».

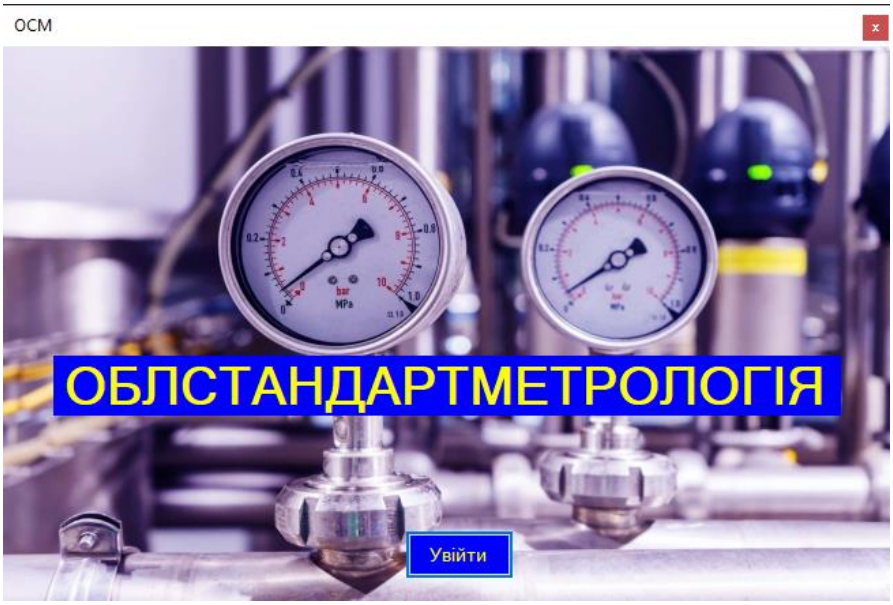


Рисунок 14 – Заставка програми (рисунок виконано самостійно)

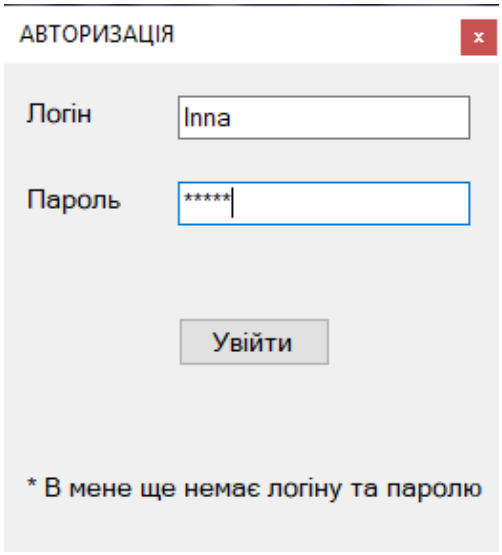


Рисунок 15 – Зовнішній вид вікна «Авторизація» (рисунок виконано самостійно)

Якщо було введено вірний логін та пароль адміністратору, то вікно авторизації буде закрито, а на екран виводиться вікно «Адміністрування».

У випадку, коли користувача ще не зареєстровано в системі, можна натиснути на надпис: «*У мене немає акаунту». Після цього буде виведене повідомлення, що для реєстрації треба звернутися до адміністратора.

У випадку, коли було введено невірний логін чи пароль, після натискання кнопки «Увійти» буде виведено повідомлення, що введено невірний логін чи пароль, та значення кількості спроб вводу логіну та паролю що залишились.

Якщо користувачу протягом трьох спроб не вдалося ввести вірні логін та пароль, то виводиться повідомлення про це. Після закриття повідомлення програма автоматично завершує свою роботу.

Владка «Співробітники» наведена на рисунку 16. У цьому режимі роботи програми адміністратор має змогу проводити облік співробітників по відділах, пошук співробітників відділу по прізвищу та фільтрування за наявністю логіну та паролю співробітника.

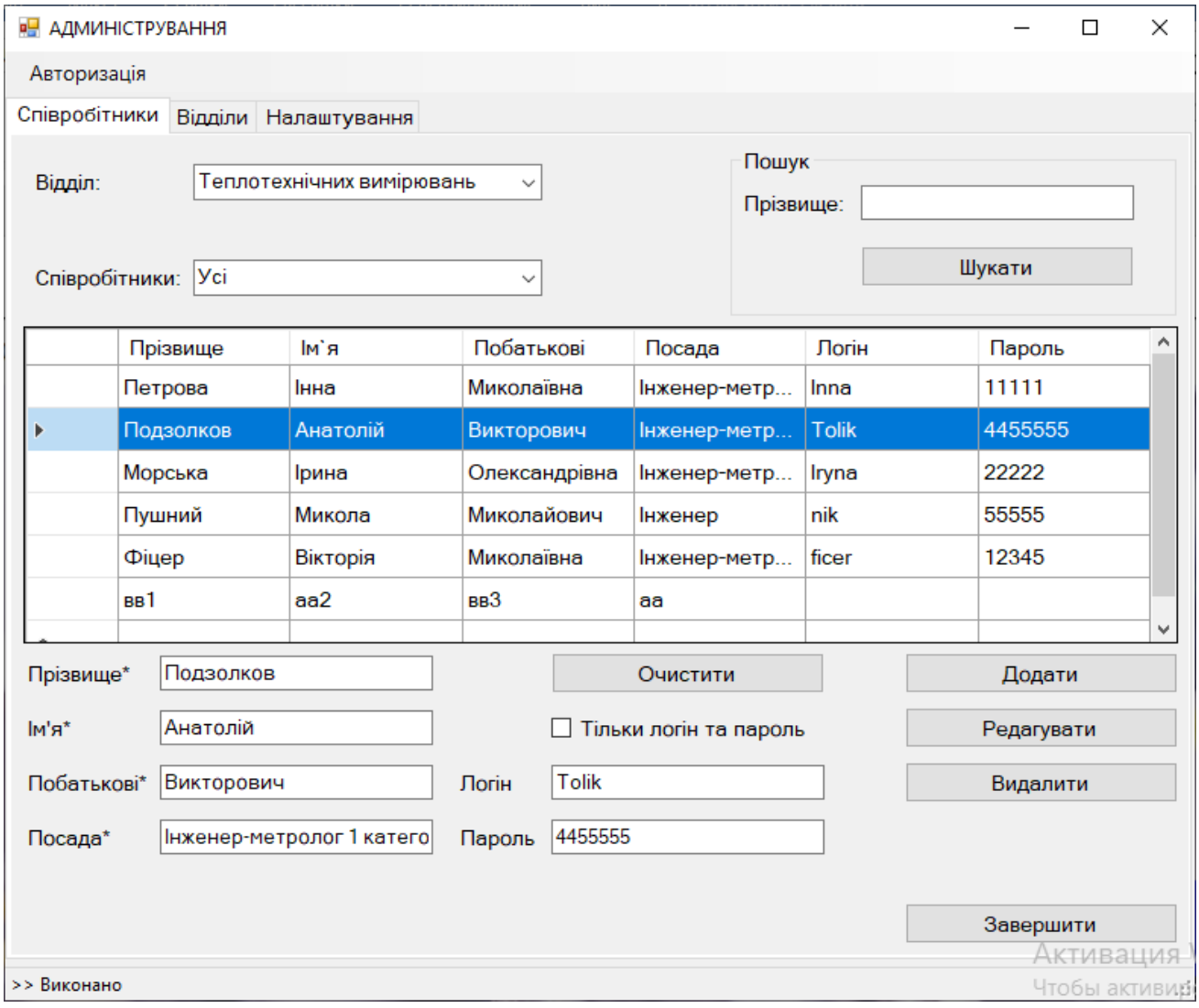


Рисунок 16 – Вікно «Адміністрування», вкладка «Співробітники »

(рисунок виконано самостійно)

На рисунку 17 наведено вкладну «Відділи», яка дозволяє додавати, редагувати та видаляти дані відділів. Меню «Налаштування» призначене для того, щоб змінити колір фону всього додатку та логін і пароль адміністратора.

4.6 Опис користувацького інтерфейсу користувача

При введені валідного значення логіну та паролю користувача відкривається головне вікно програми.

На головній формі програми відбувається первинна реєстрація замовників, оформлення замовлень, додавання приладів та таке інше. Логіка навігації по функціоналу системи була розроблена в розділі 3.4. Відповідно до неї організовано меню системи та переходи між формами додатку.

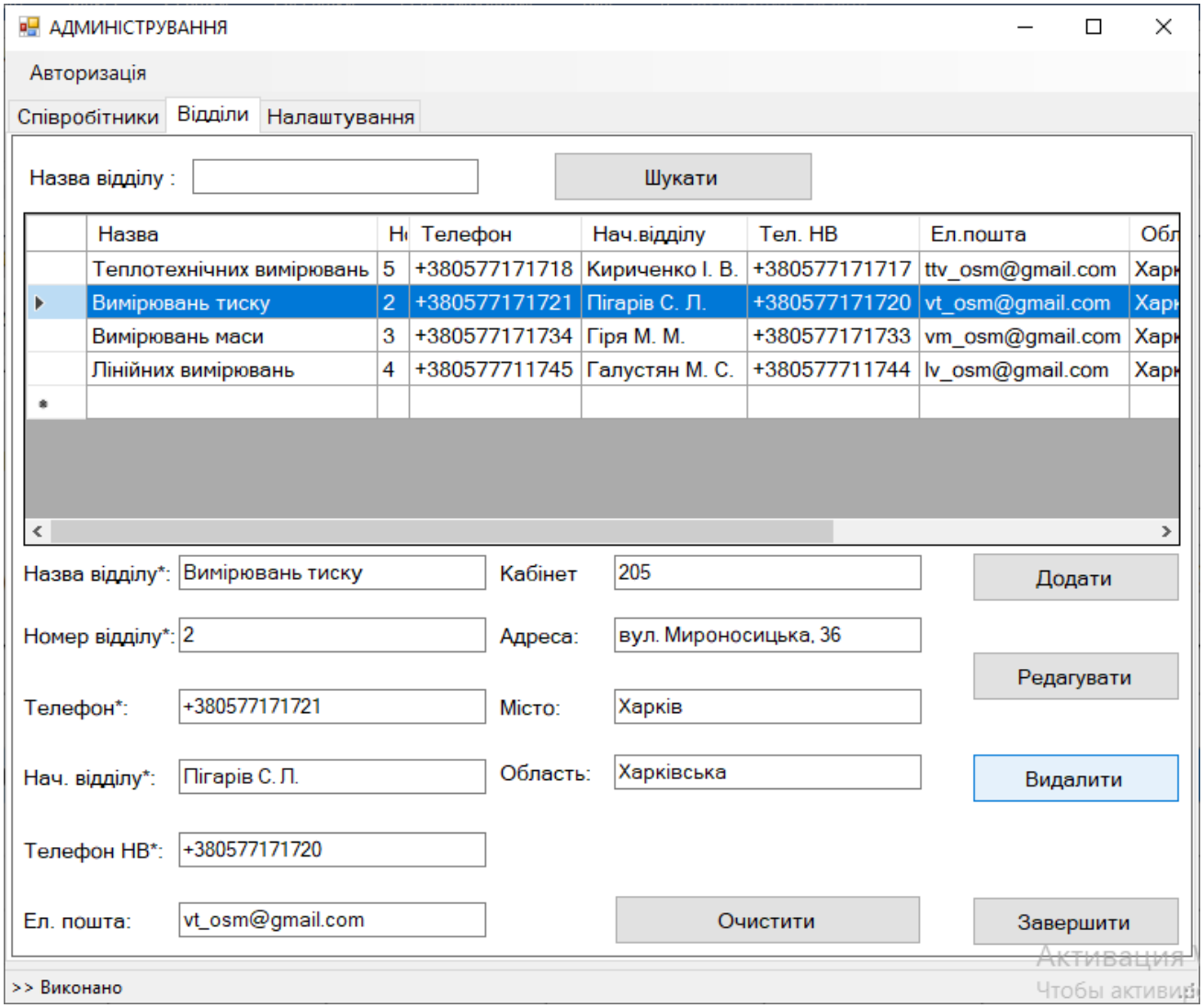


Рисунок 17 – Вікно «Адміністрування», вкладка «Відділи» (рисунок виконано самостійно)

З головної форми додатку також надано доступ до роботи з довідниковими таблицями системи: категорії приладів; прилади; параметри.

Також реалізовано доступ до статистик відповідно до роботи організації.

Отже, головним вікном програми є форма «Замовлення» (див. рис. 18), звідки починається весь процес обробки замовлень, що поступають у вигляді листа на пошту організації.

Для зручності роботи на формі реалізовано можливості:

- пошуку замовлень за номером замовлення або за номером листа;
- фільтрації інформації про наявні замовлення по статусу замовлення, по назві замовника та інтервалам дат, коли було отримано замовлення;
- сортування по будь-яким атрибутам замовлення.

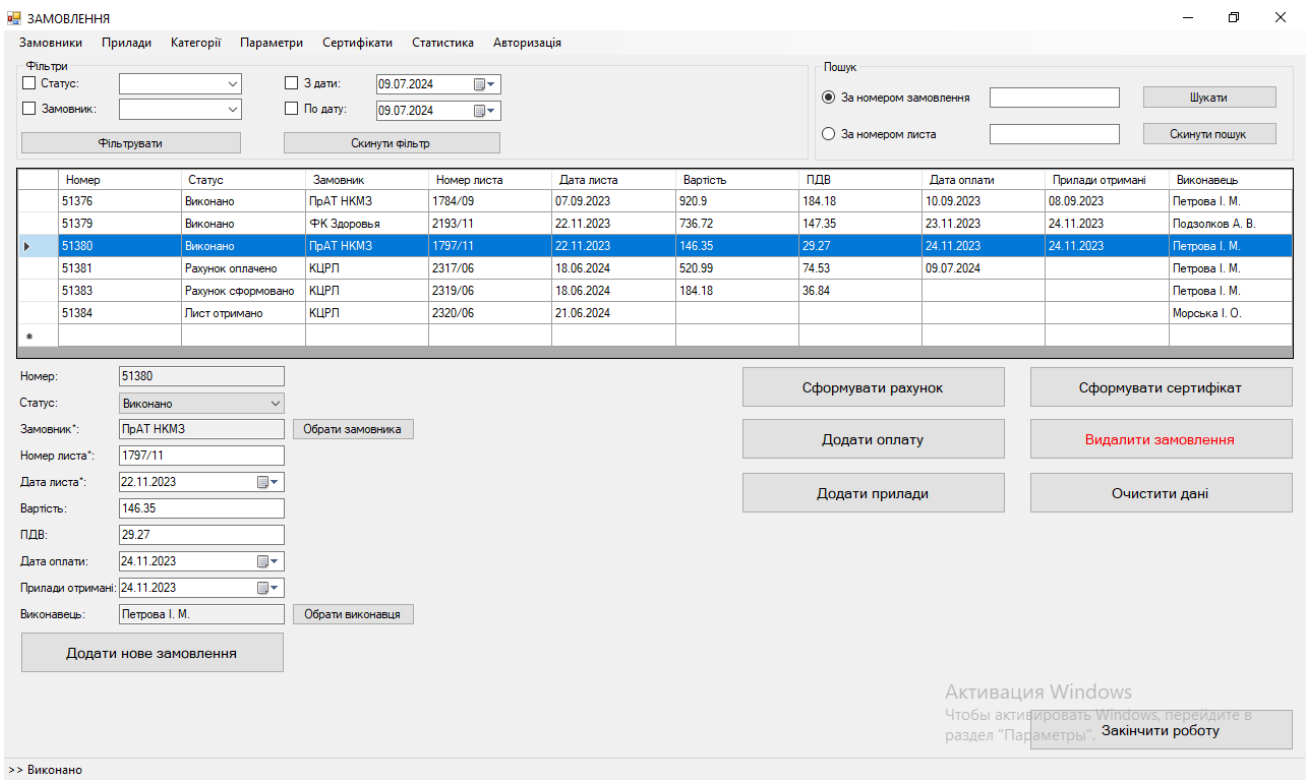


Рисунок 18 – Головне вікно програми (рисунок виконано самостійно)

При появі нового замовлення необхідно обрати замовника, який робить замовлення. Для цього треба натиснути на кнопку «Обрати замовника», після

чого буде відкрито вікно для роботи з замовниками (див. рис. 19), де можна або обрати вже існуючого замовника, або додати нового . Наприклад для додавання замовника в базу даних достатньо заповнити поля в нижній частині форми та натиснути кнопку додати. В цьому режимі можна зробити пошук замовників за назвою підприємства або по коду ЕДРПОУ. Фільтрація замовників відбувається за датою реєстрації та за містом розташування.

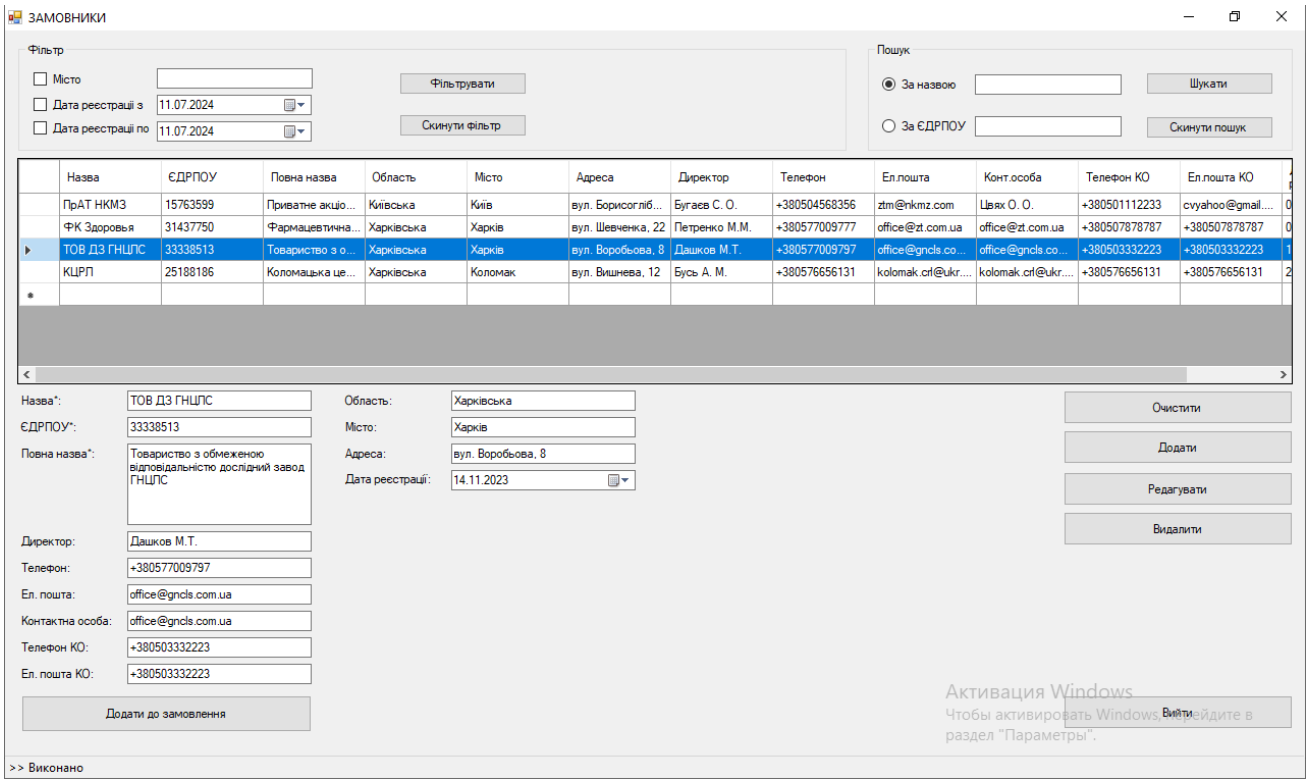


Рисунок 19 – Вікно роботи з замовниками (рисунок виконано самостійно)

Після того, як замовлення буде сформовано, необхідно додати до нього категорії приладів та їх кількість, яку замовних хоче надіслати на повірку. Для цього треба перейти в режим формування рахунку-фактури (див. рис. 20).

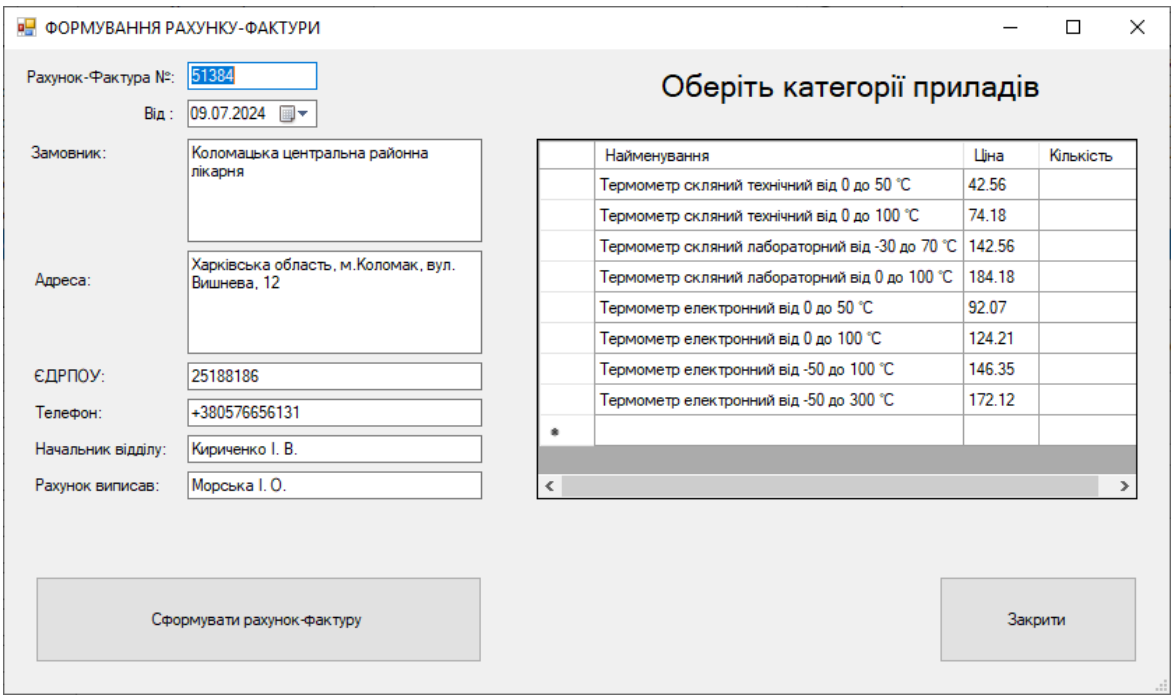


Рисунок 20 – Вікно формування рахунку-фактури (рисунок виконано самостійно)

Отже, для цього треба обрати відповідну категорію приладу та вказати кількість приладів, що буде повірятися. Після цього натиснути на кнопку «Сформувати рахунок-фактуру», після чого буде сформовано відповідний документ.

На рисунку 21 наведено приклад документу «Рахунок-фактура», якій містить перелік категорій приладів, що будуть повірятися та вартість роботи. Даний документ пересилається на пошту замовнику, який повинен його оплатити.

РАХУНОК – ФАКТУРА

№ 51381

від 06.07.2024

Виконавець:
Державне підприємство Обласний регіональний науково-виробничий центр стандартизації, метрології та сертифікації (ДП «Облстандартметрологія»).
Адреса: 61002 Україна, м. Харків, вул. Мироносицька, 26
Р/р: 2601003010403808 в АТ «Ощадбанк»
МФО: 352134
ЄДРПОУ: 04725906

Замовник:
Коломацька центральна районна лікарня
Адреса: Харківська область, м.Коломак, вул. Вишнева, 12
ЄДРПОУ: 25188186
Телефон: +380576656131

За проведення періодичної або позачергової повірки ЗВТ

№ п/п	Найменування приладів	Кількість, шт.	Ціна послуги, грн.	Вартість, грн.
1	Термометр скляний технічний від 0 до 100 °С	2	74.18	148.36
2	Термометр електронний від 0 до 100 °С	3	124.21	372.63
Вартість послуг, грн				520.99
Крім того ПДВ, грн				74.53

Загальна сума, що підлягає оплаті: 595 грн. 52 коп.

М.П.

Начальник ВТТВ _____ Кириченко І. В.
(підпис) (П.І.Б.)

Рахунок-фактуру виписав(ла) _____ Петрова І. М.
(підпис) (П.І.Б.)

Рисунок 21 — Звіт «Рахунок-фактура» (рисунок виконано самостійно)

Після того, як рахунок-фактура буде сплачений, замовник надсилає прилади для повірки, і виконавець повинен додати їх до бази даних. На рисунку 22 наведено приклад форми, на якій відбувається додавання реальних приладів для повірка.

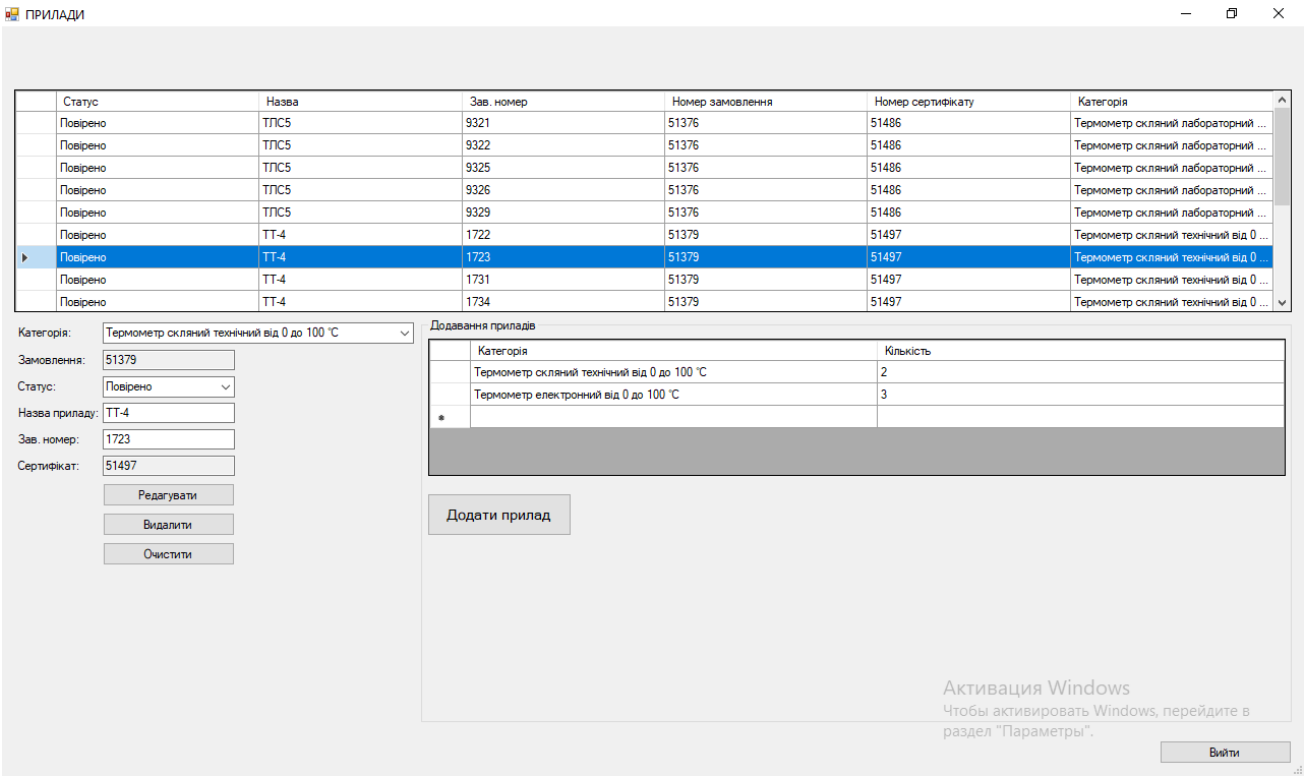


Рисунок 22 — Форма роботи з приладами (рисунок виконано самостійно)

Далі відбувається процес повірки приладів, під час якого виконавець може змінити статус приладу, вказавши що він повірений або що на нього виписаний сертифікат. На рисунку 23 наведено форму, за допомогою якою проглядається інформація до формується сертифікат.

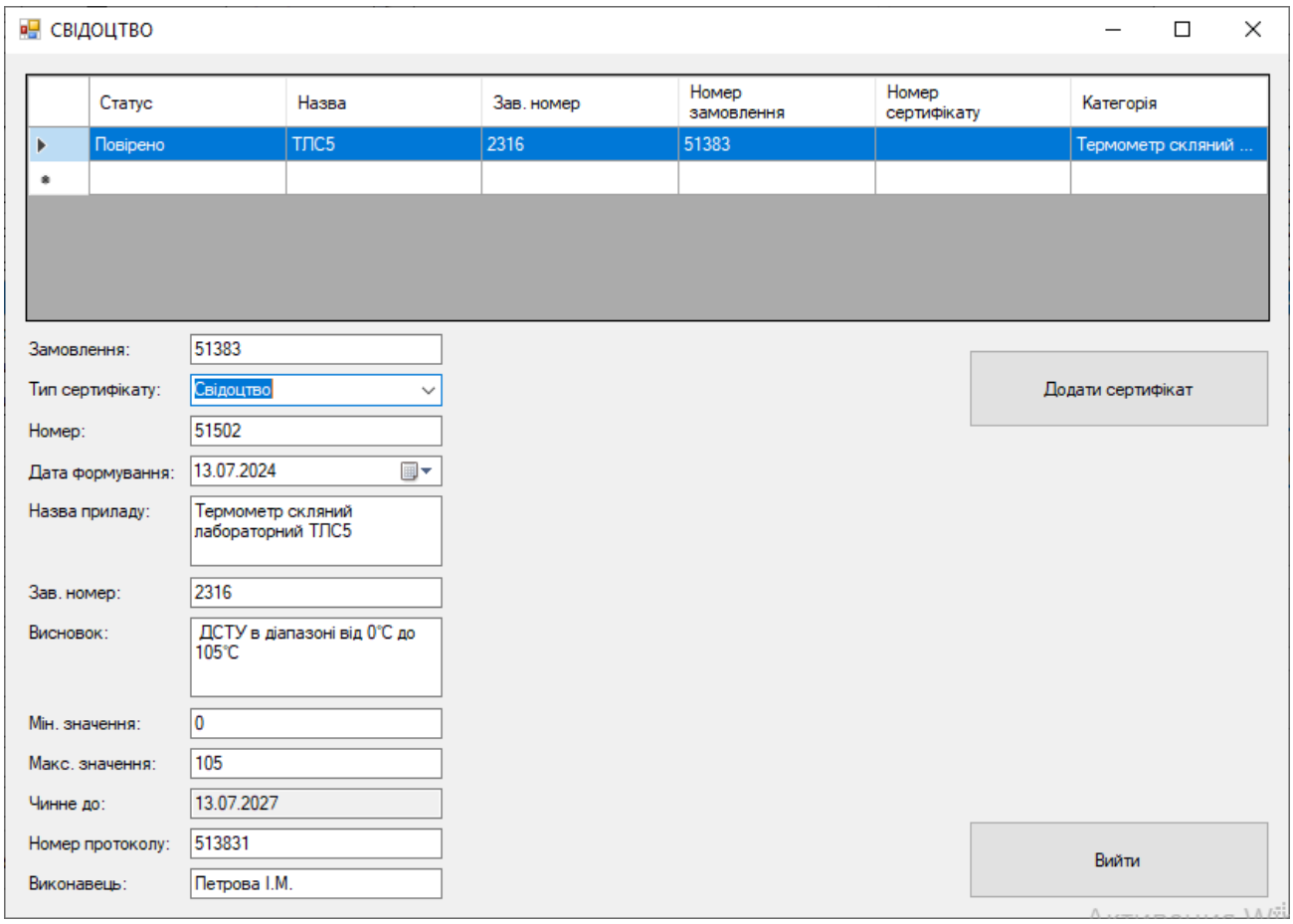


Рисунок 23 — Приклад форми «Свідоцтва» (рисунок виконано самостійно)

На рисунку 24 наведено приклад сертифікату, який може бути сформований системою по бажанню виконавця.

Після того, як всі прилади будуть повірено, тригер, що спрацьовує на серверній стороні системи, переведе замовлення в статус «Прилади повірено». А після того, як на всі прилади буде сформовано сертифікати, інший тригер переведе замовлення в статут «Сертифікати сформовано».

В разі, якщо прилад пройшов перевірку та були виявлені відхилення від норм, на цей прилад не може бути виписано свідоцтво про повірку (або сертифікат).

МІНЕКОНОМРОЗВИТКУ УКРАЇНИ
ДЕРЖАВНЕ ПІДПРИЄМСТВО
«ОБЛАСНИЙ РЕГІОНАЛЬНИЙ НАУКОВО-ВИРОБНИЧИЙ ЦЕНТР СТАНДАРТИЗАЦІЇ, МЕТРОЛОГІЇ ТА СЕРТИФІКАЦІЇ»
/ ДП «ОБЛСТАНДАРТМЕТРОЛОГІЯ»/

Уповноважено Мінекономрозвитку України.
Свідоцтво про уповноваження № ПК 007-2021 від 16.05.2021 р.

С В І Д О Ц Т В О
про повірку законодавчо регульованого засобу вимірювальної техніки

№ *51499*

Чинне до «*13*» *07 2025* р.

Назва та умовне позначення *Термометр електронний ТРМ-10-70*
Заводський номер *19235*

За результатами повірки встановлено, що засіб вимірювальної техніки відповідає вимогам *ДСТУ в діапазоні від 0°С до 100°С*

Особливості застосування засобу вимірювальної техніки *Немає*

Повірку виконав _____ *Петрова І.М.*
(підпис) (ПІБ)

Місце відбитку
повірочного тавра

«*13*» *7 2024 р.*

Рисунок 24 — Приклад сертифікату (рисунок виконано самостійно)

В цьому разі виконавець формує довідку про неуспішні результати перевірки (див. рис. 25).

Після передачі сертифікатів та повірений приладів замовнику виконавець може перевести замовлення в статус «Виконано», і на цьому життєвий цикл обробки замовлення завершується.

МІНЕКОНОМРОЗВИТКУ УКРАЇНИ
ДЕРЖАВНЕ ПІДПРИЄМСТВО
«ОБЛАСНИЙ РЕГІОНАЛЬНИЙ НАУКОВО-ВИРОБНИЧИЙ ЦЕНТР СТАНДАРТИЗАЦІЇ, МЕТРОЛОГІЇ ТА СЕРТИФІКАЦІЇ»
/ ДП «ОБЛСТАНДАРТМЕТРОЛОГІЯ»/

Уповноважено Мінекономрозвитку України.
Свідоцтво про уповноваження № ПК 007-2021 від 16.05.2021 р.

Д О В І Д К А

про повірку законодавчо регульованого засобу вимірювальної техніки

№ *30*

Чинне до «*13*» *07* *2027* р.

Назва та умовне позначення *Термометр скляний технічний ТТЖ-М1П*
Заводський номер *1124*

За результатами повірки встановлено, що засіб вимірювальної техніки не відповідає вимогам *ДСТУ в діапазоні від 0°С до 100°С*

Повірку виконав _____ *Петрова І.М.*
(підпис) (ПІБ)

М. П.

«*13*» *7* *2024 р.*

Рисунок 25 — Приклад довідки (рисунок виконано самостійно)

В системі реалізовано додаткові можливості роботи з довідниковими таблицями, де користувач може продивитися або відредагувати чи додати інформацію про:

- категорії приладів;
- параметри приладів;
- тощо.

Система дозволяє отримувати статистики, які потрібні начальникам відділів щодо продуктивності роботи співробітників та тому подібне.

На рисунку 26 наведено результат отримання статистики про суми, які заробити співробітники певного відділу за певний часовий період.

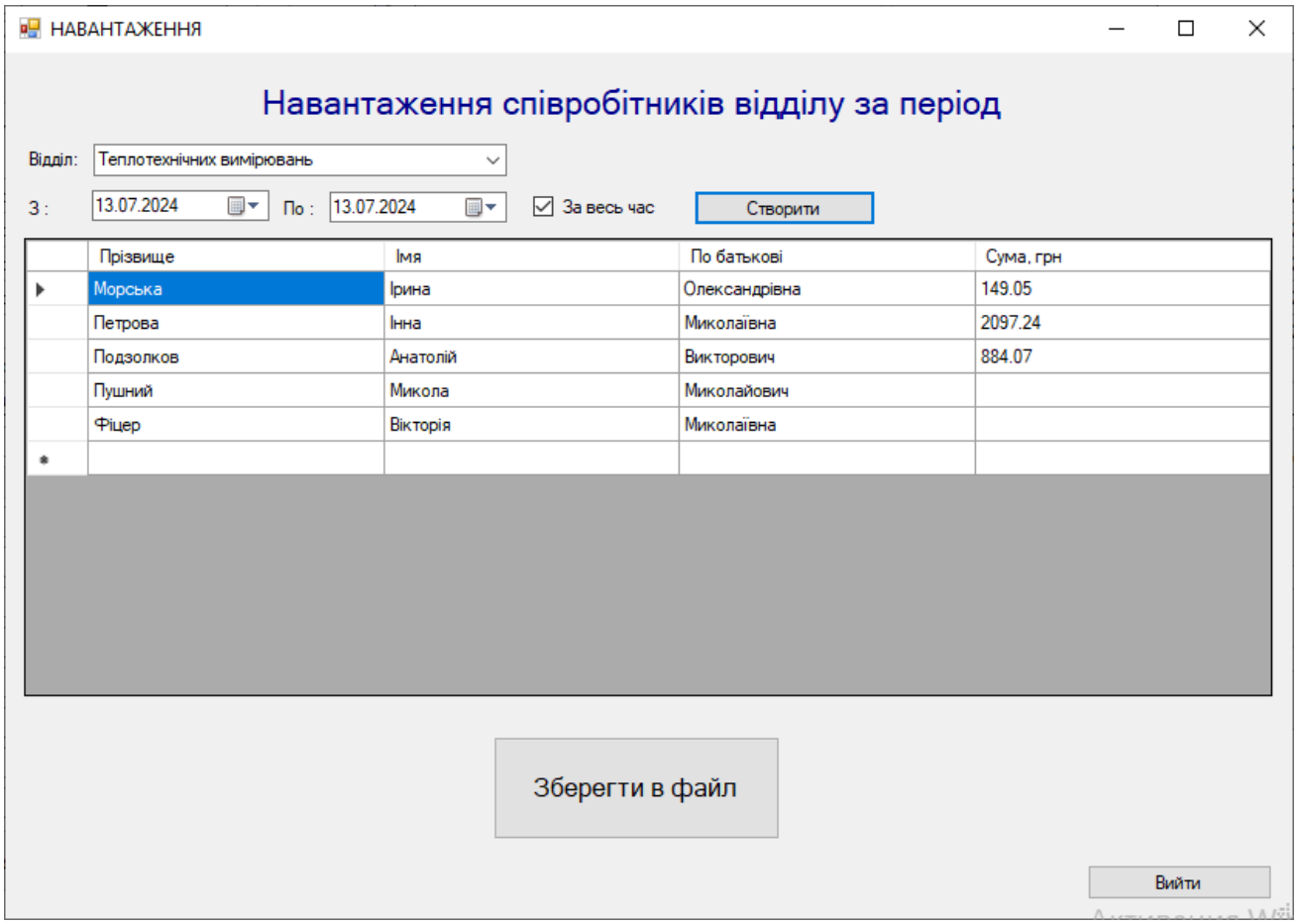


Рисунок 26 — Статистика щодо продуктивності співробітників відділу (рисунок виконано самостійно)

Дана статистика отримана за допомогою наступного запиту.

```
queryString1 = $"SELECT E.Ex_surname as 'Прізвище', " +
        $"E.Ex_name as 'Ім'я', " +
        $"E.Ex_patname as 'По батькові', " +
        $"ROUND(SUM(O.Ord_cost + O.Ord_VAT),2) as 'Сума, грн' " +
        $"FROM Executor E, Orders O " +
        $"WHERE E.Dep_id = '{comboBox_Dep.SelectedValue.ToString()}' "
+
        $"AND E.Ex_id = O.Ex_id " +
        $"AND                    O.Ord_let_date                    BETWEEN
DATEFROMPARTS({dt1.Year},{dt1.Month},{dt1.Day}) " +
        $"AND DATEFROMPARTS({dt2.Year},{dt2.Month},{dt2.Day})" +
        $"GROUP BY E.Ex_surname, E.Ex_name, E.Ex_patname " +
```

```
$"UNION " +
$"SELECT E.Ex_surname, E.Ex_name, E.Ex_patname, NULL " +
$"FROM Executor E " +
$"WHERE E.Ex_id NOT IN (SELECT DISTINCT O.Ex_id " +
$"FROM ORDERS O) " +
$"AND E.Dep_id = '{comboBox_Dep.SelectedValue.ToString()}';";
```

На рисунку 27 наведено приклад статистики по кількості приладів, що повірялися в організації, за різними категоріями.

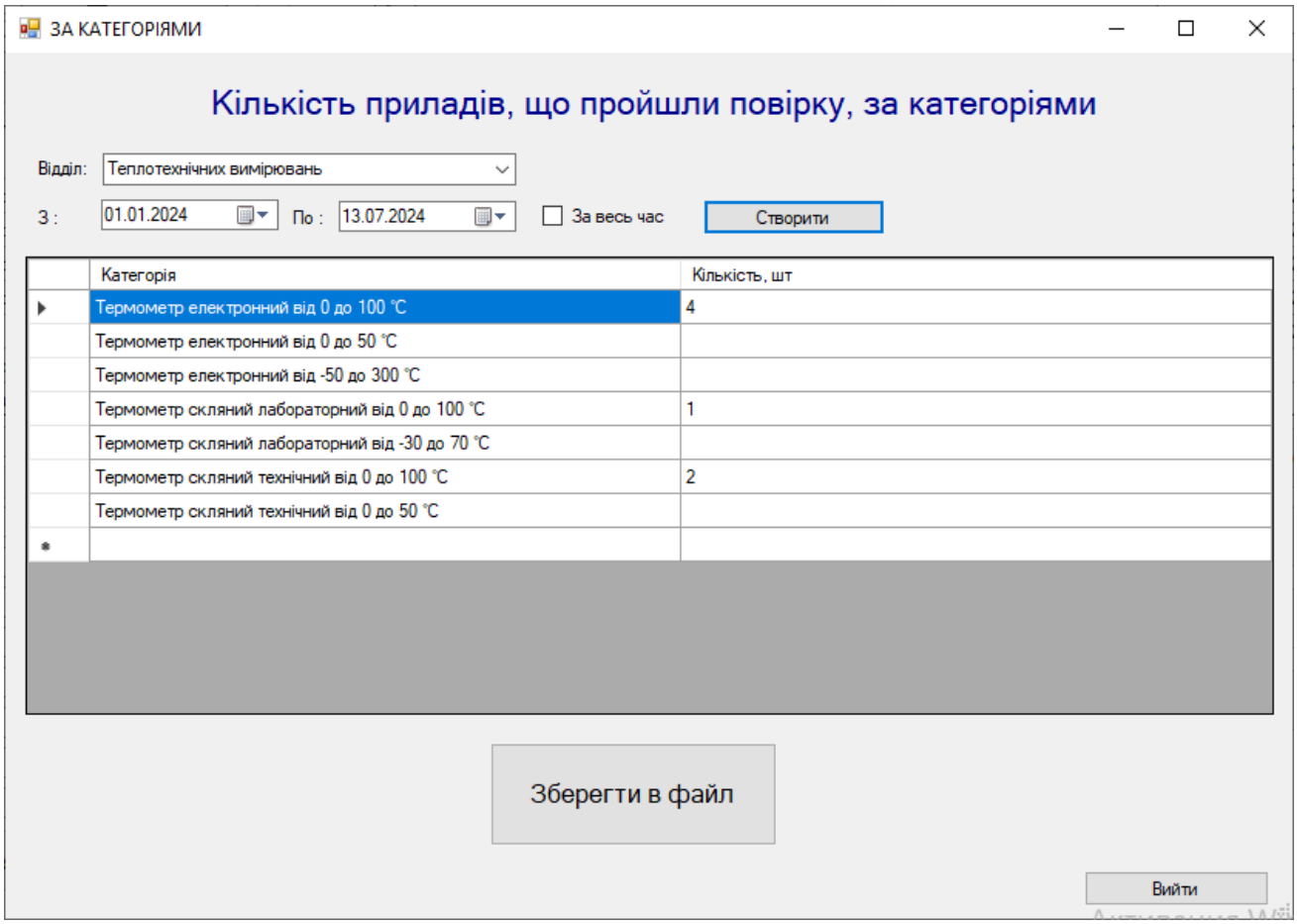


Рисунок 26 — Статистика проведених повірок по категоріях приладів вимірювальної техніки (рисунок виконано самостійно)

Дана статистика отримана за допомогою наступного запиту.

```
queryString1 = $"SELECT C.Cat_name as 'Категорія', " +
      $"SUM(P.Pos_quantity) as 'Кількість, шт' " +
      $"FROM Category C, Position P, Orders O " +
      $"WHERE P.Cat_id = C.Cat_id " +
```

```
            $"AND P.Ord_id = O.Ord_id " +
            $"AND                    O.Ord_let_date                    BETWEEN
DATEFROMPARTS({dt1.Year},{dt1.Month},{dt1.Day})" +
            $"AND DATEFROMPARTS({dt2.Year},{dt2.Month},{dt2.Day}) " +
            $"GROUP BY C.Cat_name " +
            $"UNION " +
            $"SELECT C.Cat_name, NULL " +
            $"FROM Category C " +
            $"WHERE C.Cat_id NOT IN (SELECT DISTINCT P.Cat_id " +
            $"FROM Position P);";
```

Всі статистики можуть бути не тільки проглянуті на відповідних формах, але і збережені в Word-документах для подальшого друку та звітності.

Система дозволяє також формувати інші звіти.

Скрипти на формування звітів та інші функції системи наведено у додатку В.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обґрунтування вибору виду тестування

Важливим етапом життєвого циклу програмної системи є її тестування. Існує велика кількість методів тестування, які належать до певних категорій. Так, за рівнем тестування можна виділити наступні підходи до тестування: модульне, інтеграційне, системне тестування.

Враховуючі архітектуру програмної системи, що розроблено, необхідно провести окремо тестування серверної частини системи та тестування клієнтської частини програмної системи.

Для обох частин було обрано модульне тестування. Модульне тестування – це тестування, якє передбачає тестування окремо кожного модуля з програмного коду програмної системи. Модуль розглядається як найменша частина програми, на якій можна провести тестування. Отже, тестуванню підлягає кожен метод та функція програми.

Також для тестування програмної системи можна розглянути такі види тестування як функціональне тестування та тестування інтерфейсу.

Функціональне тестування — це тестування, що пов’язане з перевіркою відповідності функціональності програми до тих вимог, що задано в специфікації системи. Мета функціонального тестування - виявлення розбіжностей між поведінкою функцій, яка очикується за специфікацією, та реальною їх роботою. Тестові сценарії, які розробляються для функціонального тестування, повинні охопити всі функції, що реалізовано в системі, з урахуванням всіх найбільш вірогідних помилок.

Тестування інтерфейсу користувача — це тестування, що пов’язане з перевірками правильності роботи інтерфейсу користувача також відповідно до специфікацій. Під час такого тестування можуть бути виявлені такі типи помилок:

- помилки в графічному інтерфейсі користувача, коли реалізований інтерфейс не відповідає проектній документації або якийсь елемент інтерфейсу відсутній;
- втрата або невірний вивід даних, що передаються через ті чи інші елементи користувацького інтерфейсу;
- необроблені виключні ситуації, що виникають під час взаємодії з користувацьким інтерфейсом.

Якщо згадати про ступінь тестування, то можна виділити два ступіні тестування: ручне та автоматизоване. Ручне тестування пов’язане з перевіркою тест-кейсів без використання відповідних інструментів автоматизації, тобто вручну. Автоматизоване тестування, в свою чергу, пов’язане з використнням певного набору інструментів для автоматизованої перевірки тест-кейсів.

Під час виконання кваліфікаційної роботи було обране ручне тестування системи.

5.2 Розробка тестових випадків

Test case (тестовий сценарій) — це деталізований опис всіх вхідних даних, умов виконання певного функціоналу, послідовності дій та очікуваних результатів тієї чи іншої функції. Отже, оскільки метою тестування є виявлення максимальної кількості помилок та дефектів, то вдалим вважається такий тест, який з великим ступенем ймовірності може таку помилку виявити. Під час створення тестів тестувальник повинен підходити до самого процесу з позиції так званого «конструктивного руйнування» програмної системи, таким чином знаходячи можливі проблеми з метою їх усунення.

Тестові випадки для системи «Облстандартметрологія» були розроблені на основі функції програмної системи:

- авторизація користувачів у системі;
- додавання співробітника або відділа;
- створення замовлення на повірку;
- додавання замовника;

- формування рахунок-фактури;
- формування сертифікату на прилад;
- редагування статусу замовлення та статусу приладу.

Під час проведення тестування було виявлено, що всі розроблені тести пройшли успішно. У таблиці 5.1 наведено деякі приклади тестових випадків для системи «Облстандарметрологія».

Таблиця 5.1 – Приклади тестових випадків (таблиця створена самостійно)

№	Опис	Кроки для відтворення	Очікувані результати	Реальні результати	Статус
1	Авторизація користувача	На формі-заставці системи натиснути на кнопку Увійти, ввести валідні дані	Успішна авторизація. Відкривається або вікно адміністрування, або вікно з проглядом замовлень	Успішна авторизація. Відкрилося або вікно адміністрування, або вікно з проглядом замовлень	Успіх
2	Додавання відділа	Авторизуватись як адміністратор, перейти на вкладку Відділи, ввести валідні значення про новий відділ, натиснути кнопку Додати	Інформація про новий відділ з’явиться в таблиці на вкладці Відділи	Інформація про новий відділ з’явилася в таблиці на вкладці Відділи	Успіх

Кінець таблиці 5.1

3	Формування рахунок-фактури	На формі «Формування рахунку-фактури» внести валідні дані про кількість приладів по всім категоріям приладів та натиснути на кнопку «Сформувати рахунок-фактуру»	Відкриється форма з відображенням звіту «Рахунок - фактура», де будуть присутні всі дані, що було відображено на попередній формі, інформація про замовника та виконавця, а також буде підрахована сума, що підлягає оплаті; сформовано місце для підписів	Відкрилася форма з відображенням звіту «Рахунок - фактура», де присутні всі дані, що було відображено на попередній формі, інформація про замовника та виконавця, а також підрахована сума, що підлягає оплаті; сформовано місце для підписів	Успіх

Отже, в таблиці на наведено тестові випадки, що базуються на повсякденних функціях, що використовуються в програмній системі. Всі тест-кейси були з успішним результатом виконані декілька разів, що говорить про стабільність і надійність роботи програмної системи.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було виконано проектування програмної системи метрологічного обліку приладів, що підтримує цей процес на всіх етапах від реєстрації замовлення до виписки сертифікатів на прилади, що пройшли повірку.

Було проведено аналіз та моделювання предметної області метрології, під час якого було побудовано необхідні UML-діаграми та діаграму потоків даних.

Було спроектовано БД шляхом побудови ER-діаграми та схеми реляційної БД, яка була перевірена на відповідність 3НФ. Використано логіку проектування БД через етапи концептуального моделювання, інфологічного, логічного та створення фізичної моделі за допомогою СКБД Microsoft SQL Server та SQL Server Management Studio. Під час проектування UI/UX системи розроблено карту навігації по системі.

Розроблено клієнт-серверну архітектуру програмної системи, а також спроектовано логіку, яка буде виконувати на серверній та клієнтській стороні системи. В результаті була програмно реалізована система з метрологічного обліку приладів з використанням реляційних баз даних та логіки об’єктно-орієнтованого програмування.

Таким чином, під час виконання кваліфікаційної роботи виконано наступні задачі:

- проведено аналіз та моделювання предметної області обліку метрологічних приладів;
- спроектовано базу даних для збереження інформації з предметної області;
- розроблено алгоритми підтримки бізнес-процесу метрологічного обліку приладів;
- спроектовано архітектуру програмної системи;
- виконано програмну реалізацію серверної та клієнтської частин системи та проведено тестування створеної програмної системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Наказ № 193 від 08.02.2016 року «Про затвердження порядку проведення повірки законодавчо регульованих засобів вимірювальної техніки, що перебувають в експлуатації, та оформлення її результатів». URL: https://zakononline.com.ua/documents/show/364102___553193 (дата звернення 12.07.2023);

2. Date C.J. Database Design and Relational Theory: Normal Forms and All That Jazz. Apress, 2019. 470 P. ISBN 978-148-425-539-1. DOI: https://doi.org/10.1007/978-1-4842-5540-7;

3. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 11.04.2024);

4. Bagui S., Earp R. Database Design Using Entity-Relationship Diagrams (Foundations of Database Design). – Auerbach Publications, 2011. – 371 P. – ISBN 978-143-986-177-6;

5. Meier A., Kaufmann M. SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management. – Springer Vieweg, 2019. – 248 P. – ISBN 978-3658245481;

6. Грабер М. SQL./ М.Грабер. - К.: Вид-во “ЛОРІ”, 2003. - 644 с.;

7. BAS Громадське харчування [Електронний ресурс] – Режим доступу до ресурсу: https://www.softcom.ua/ua/bas/programs/bas-gromadske-kharchuvannya-new (дата звернення 01.04.2024);

8. Аналіз ринку громадського харчування [Електронний ресурс] – Режим доступу до ресурсу: https://most.eu.com/ua/analysis/view/analiz-rynka-obshestvennogo-pitaniya (дата звернення 10.04.2024);

9. Постанова Кабінету Міністрів України №374 від 04.06.2015 року «Про затвердження переліку категорій законодавчо регульованих засобів вимірювальної техніки, що підлягають періодичній повірці». URL:

https://zakon.rada.gov.ua/laws/show/374-2015-%D0%BF#Text (дата звернення 10.05.2024);

10. Закон України про «Метрологію та метрологічну діяльність» від 05.06.2014 року. URL: https://zakon.rada.gov.ua/laws/show/1314-18#Text (дата звернення 15.03.2024);

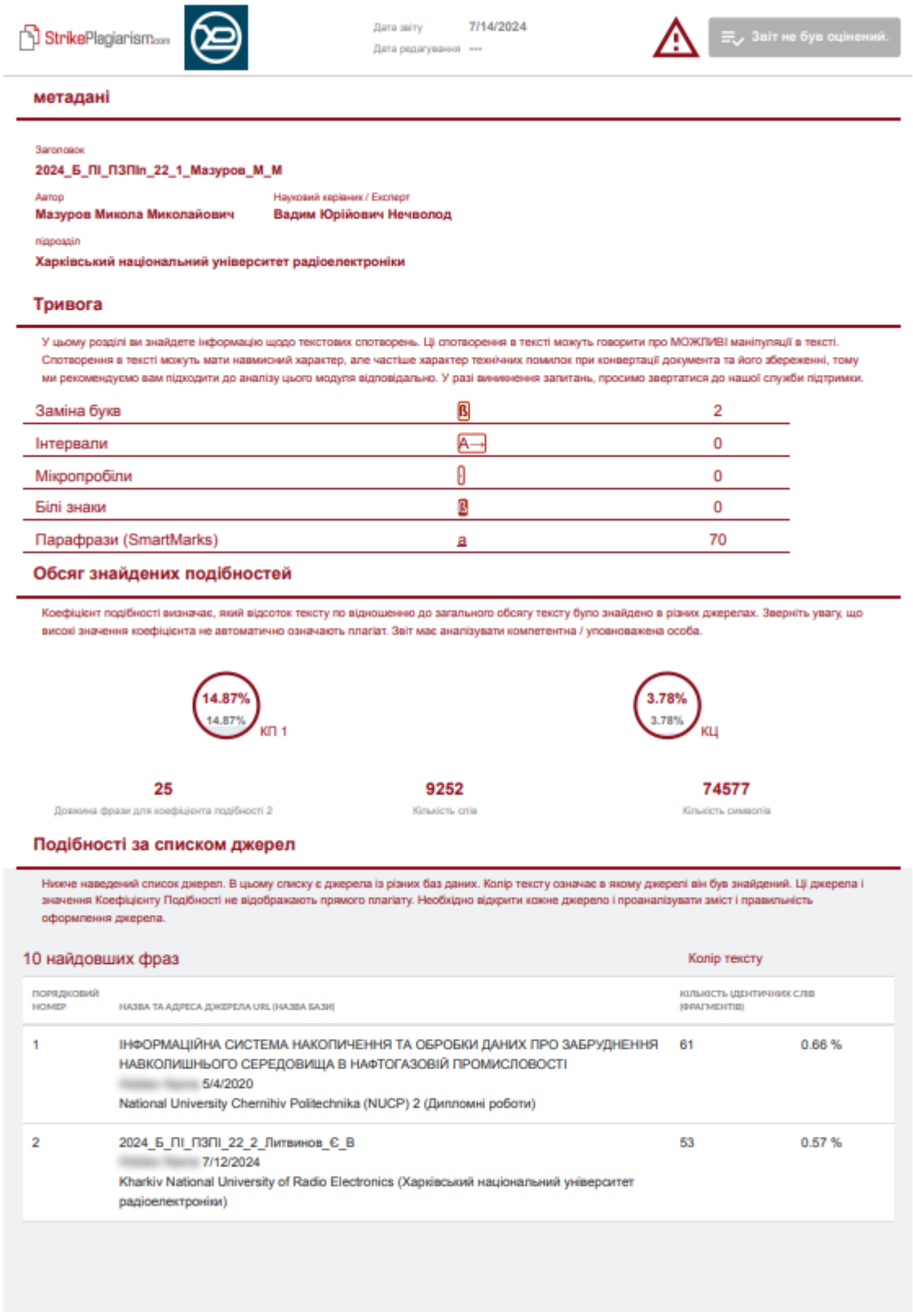
11. Керівництво з ADO.NET URL: https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ (дата звернення 29.04.24);

12. Windows Forms Microsoft документація URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0 (дата звернення 25.05.24);

13. SQL Server technical documentation URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16 (дата звернення 01.04.24).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/14/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп_22_1_Мазуров_М_М

Автор
Мазуров Микола Миколайович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		2
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		70

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

14.87% 14.87% КП 1

3.78% 3.78% КЦ

25
Довжина фрази для коефіцієнта подібності 2

9252
Кількість слів

74577
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	ІНФОРМАЦІЙНА СИСТЕМА НАКОПИЧЕННЯ ТА ОБРОБКИ ДАНИХ ПРО ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА В НАФТОГАЗОВІЙ ПРОМИСЛОВОСТІ [redacted] 5/4/2020 National University Chernihiv Politechnika (NUCP) 2 (Дипломні роботи)	61	0.66 %
2	2024_Б_ПІ_ПЗПІ_22_2_Литвинов_Є_В [redacted] 7/12/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	53	0.57 %

ДОДАТОК Б

Слайди презентації

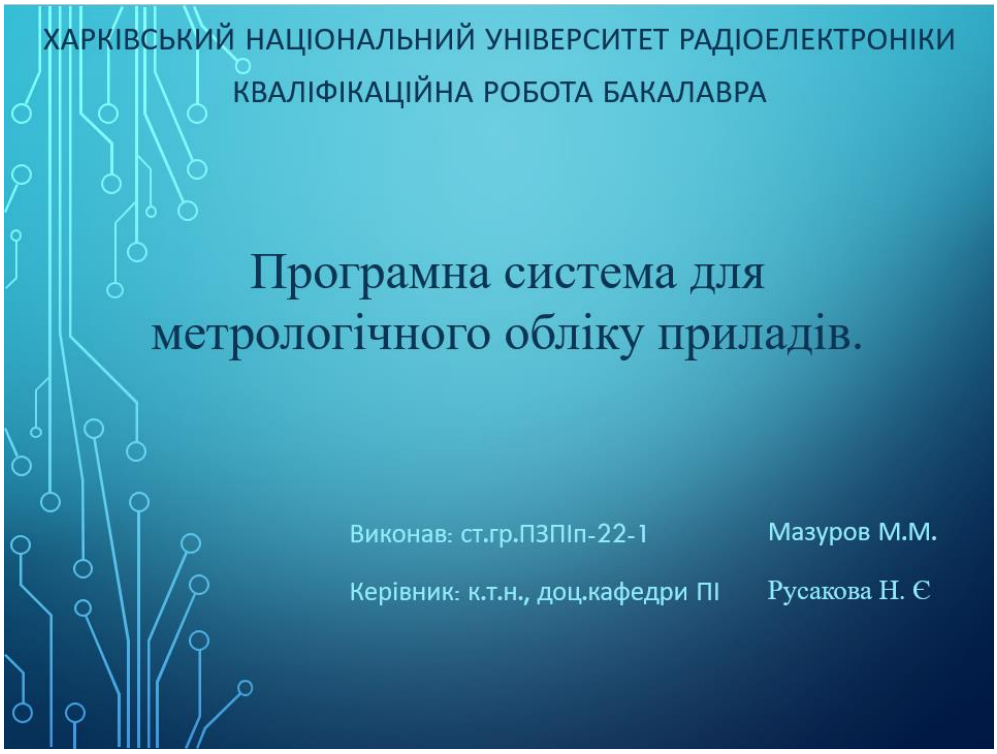


Рисунок Б.1 – Титульний слайд » (рисунок виконано самостійно)

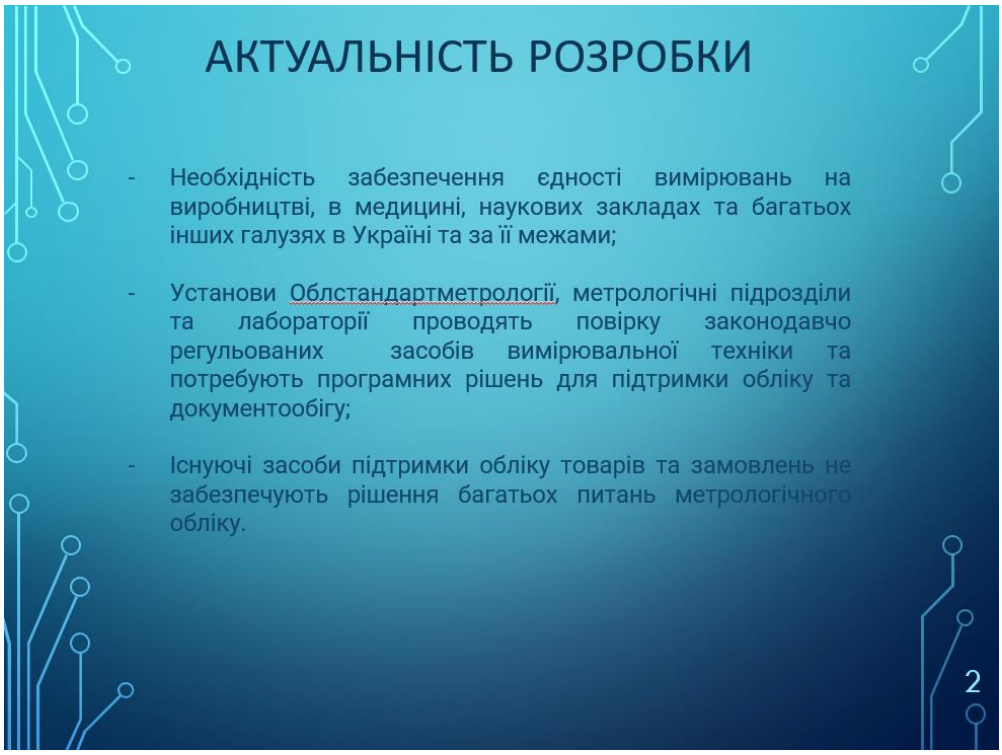


Рисунок Б.2 – Слайд «Актуальність розробки» » (рисунок виконано самостійно)

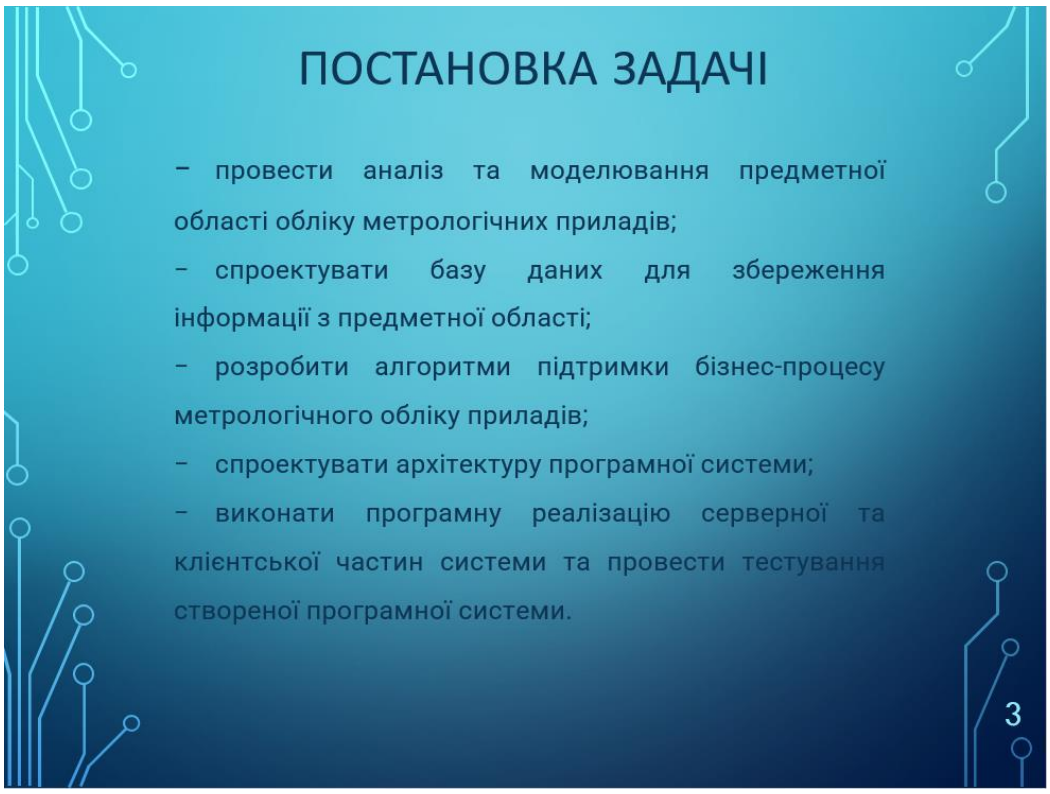


Рисунок Б.3 – Слайд «Постановка задачі» » (рисунок виконано самостійно)

Рисунок Б.4 – Слайд «Аналіз та моделювання предметної області» » (рисунок виконано самостійно)

Рисунок Б.5 – Слайд «Концептуальне моделювання» » (рисунок виконано самостійно)

Рисунок Б.6 – Слайд «Проектування бази даних» » (рисунок виконано самостійно)

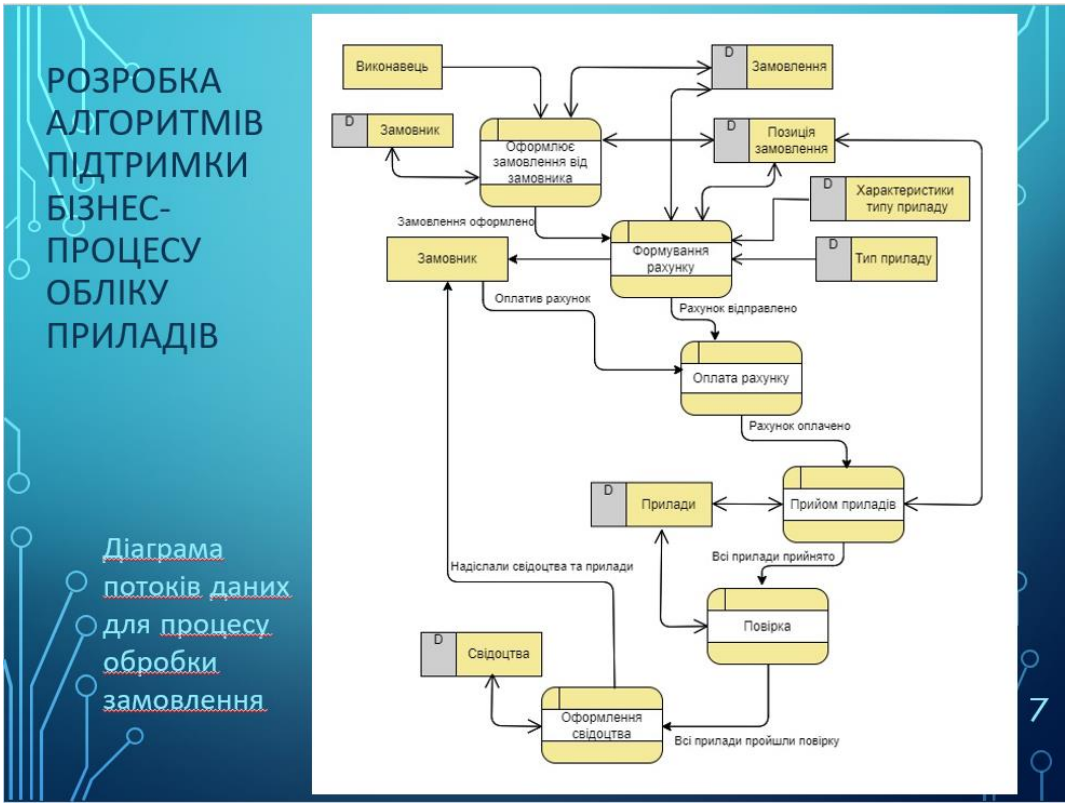


Рисунок Б.7 – Слайд «Розробка алгоритмів підтримки бізнес-процесу обліку приладів» » (рисунок виконано самостійно)

Рисунок Б.8 – Слайд «Розробка алгоритмів підтримки бізнес-процесу обліку приладів» » (рисунок виконано самостійно)

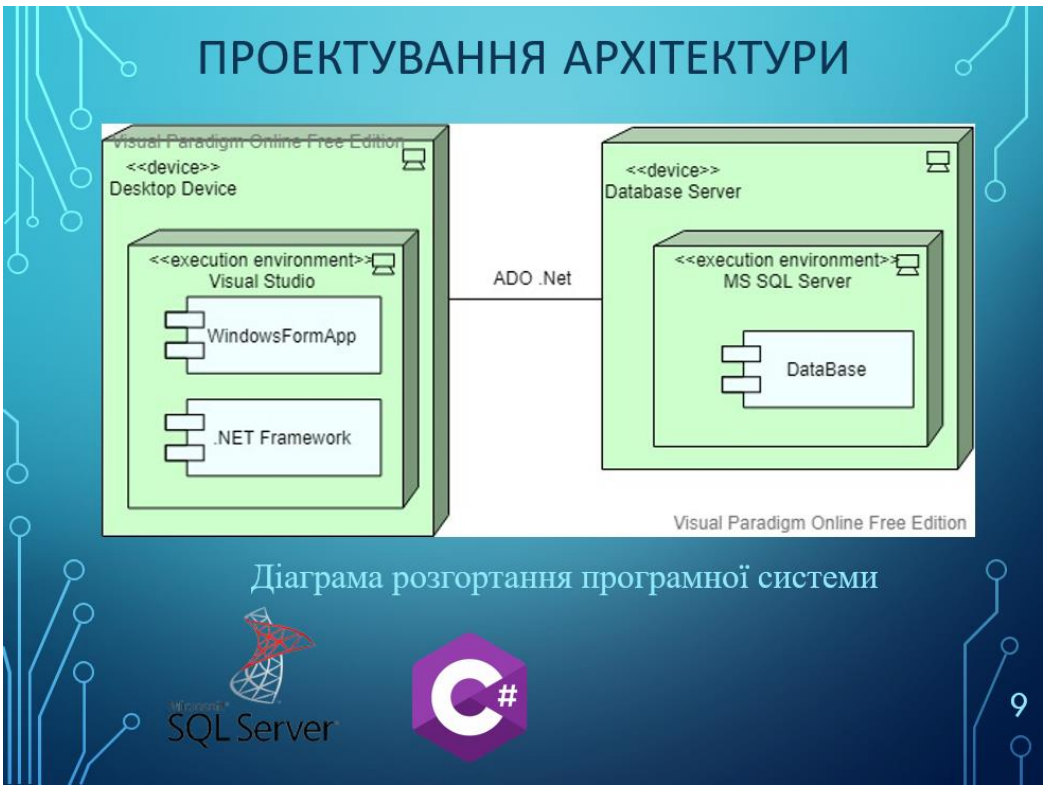


Рисунок Б.9 – Слайд «Проектування архітектури» (рисунок виконано самостійно)

Рисунок Б.10 – Слайд «Створення UI/UX дизайну системи» (рисунок виконано самостійно)

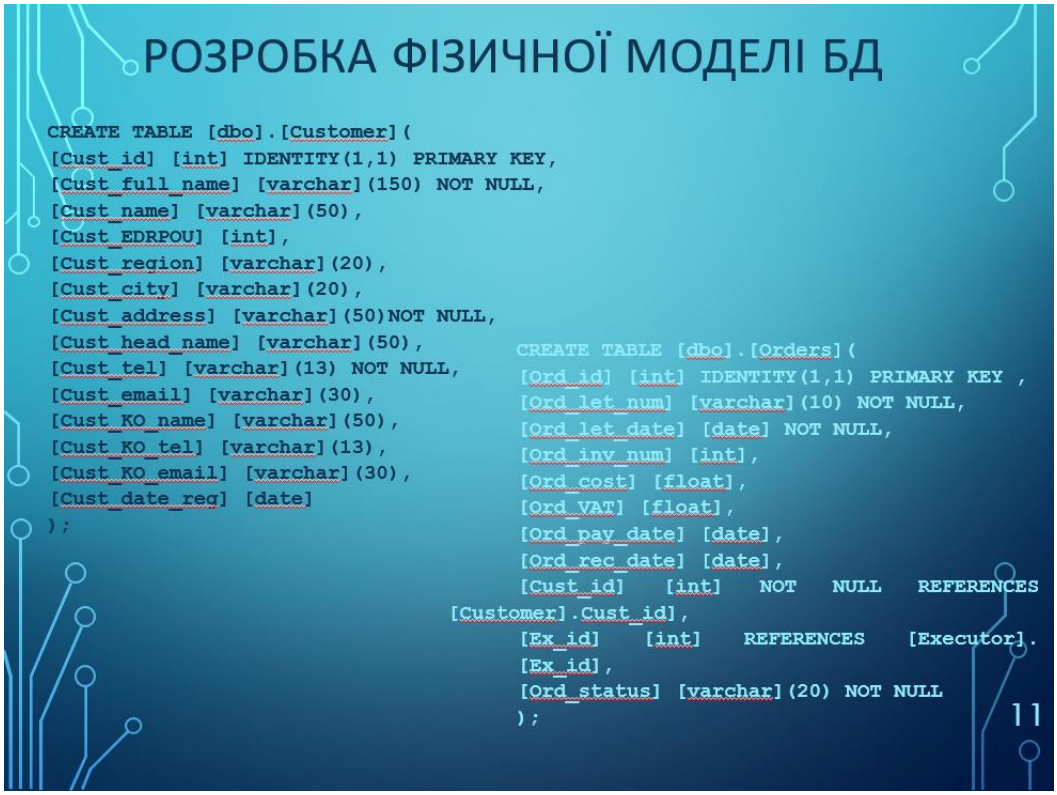


Рисунок Б.11 – Слайд «Розробка фізичної моделі БД» (рисунок виконано самостійно)

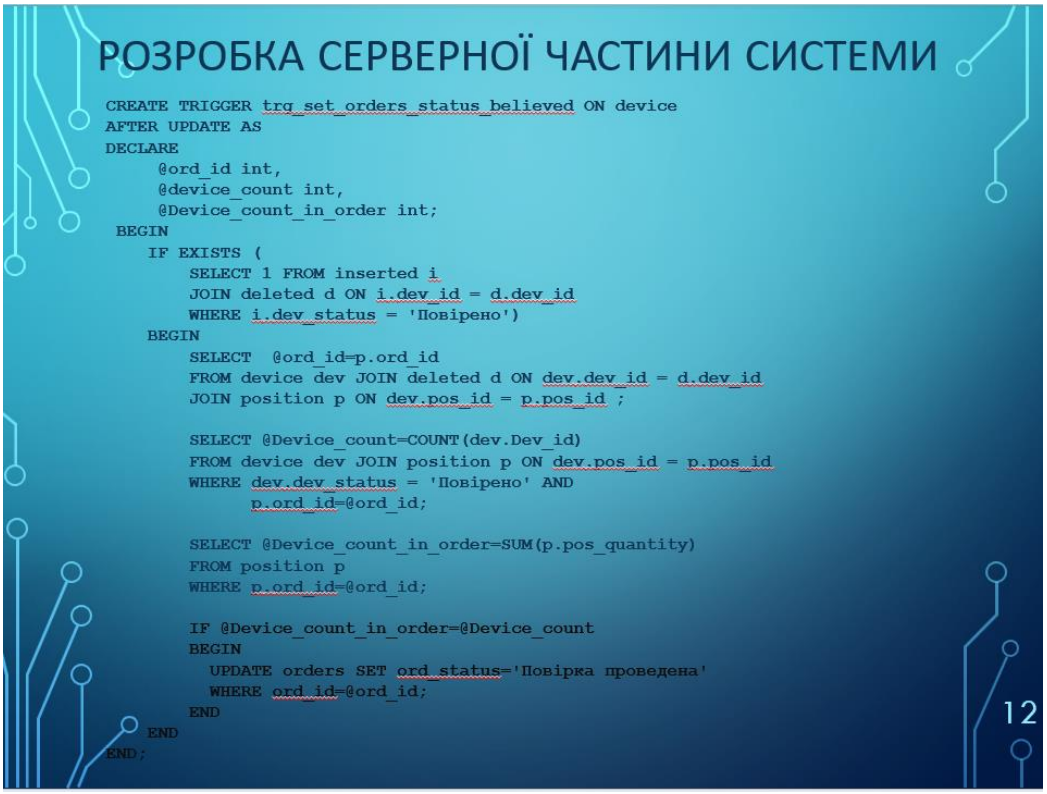


Рисунок Б.12 – Слайд «Розробка серверної частини системи» (рисунок виконано самостійно)

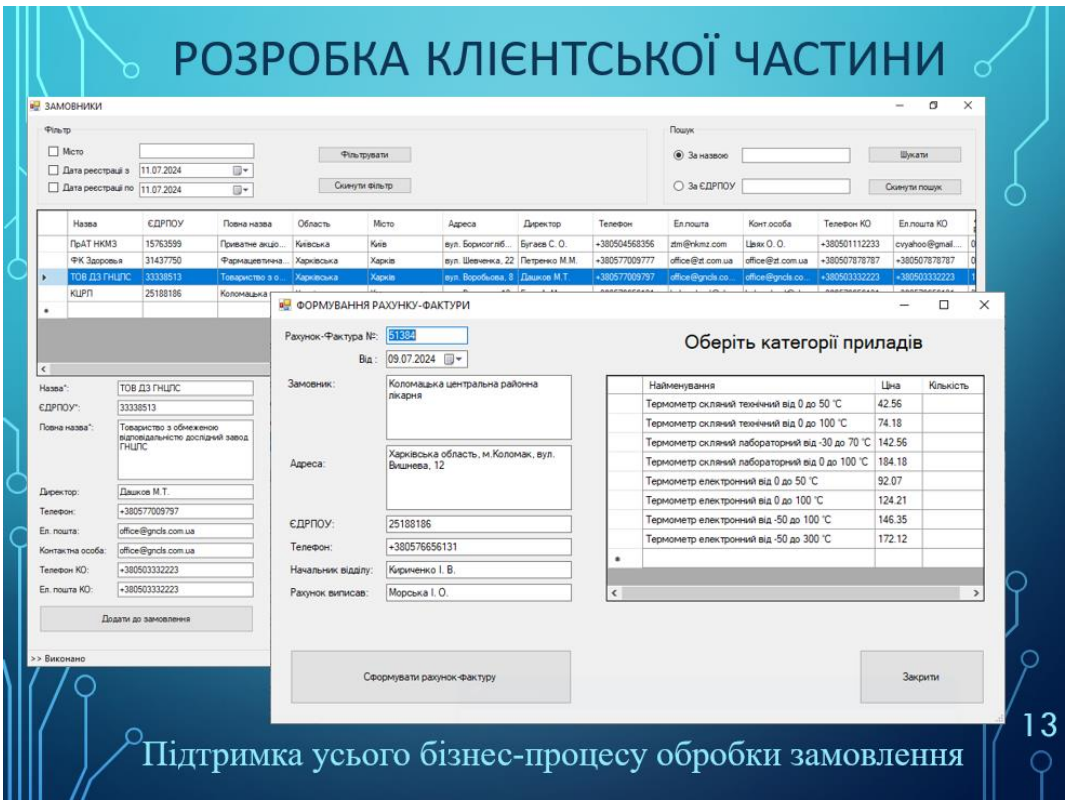


Рисунок Б.13 – Слайд «Розробка клієнтської частини» (рисунок виконано самостійно)

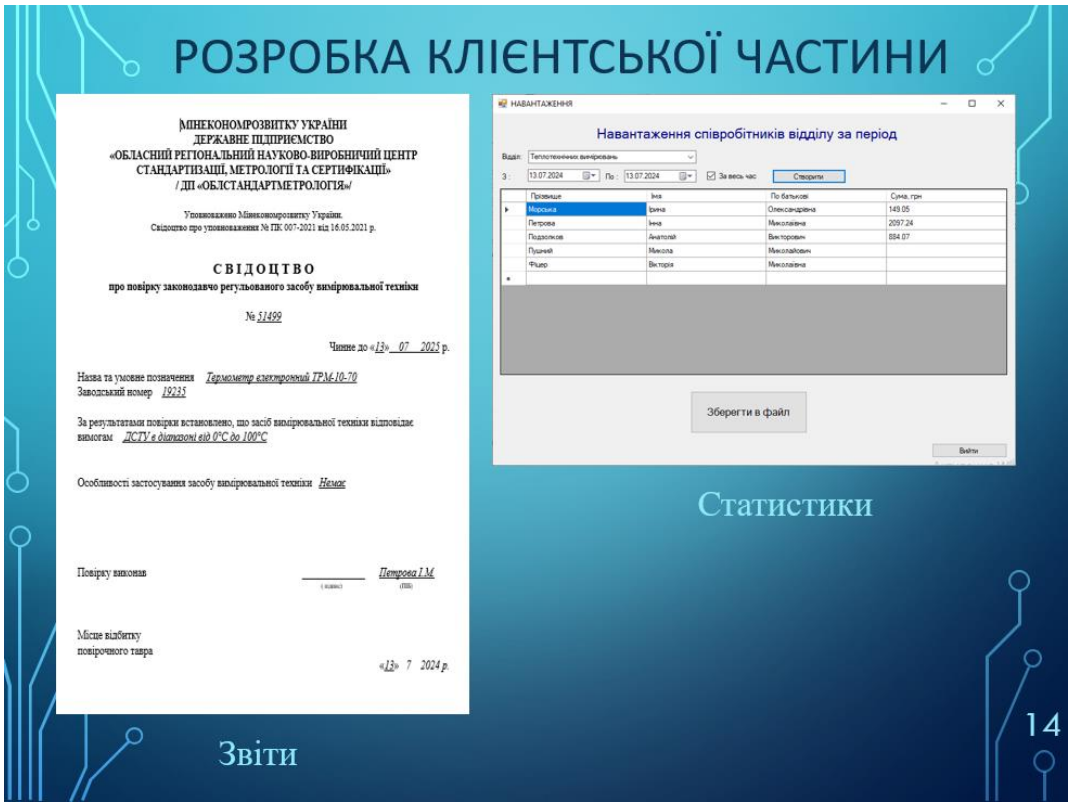


Рисунок Б.14 – Слайд «Розробка клієнтської частини» (рисунок виконано самостійно)

Рисунок Б.15 – Слайд «Тестування програми» (рисунок виконано самостійно)

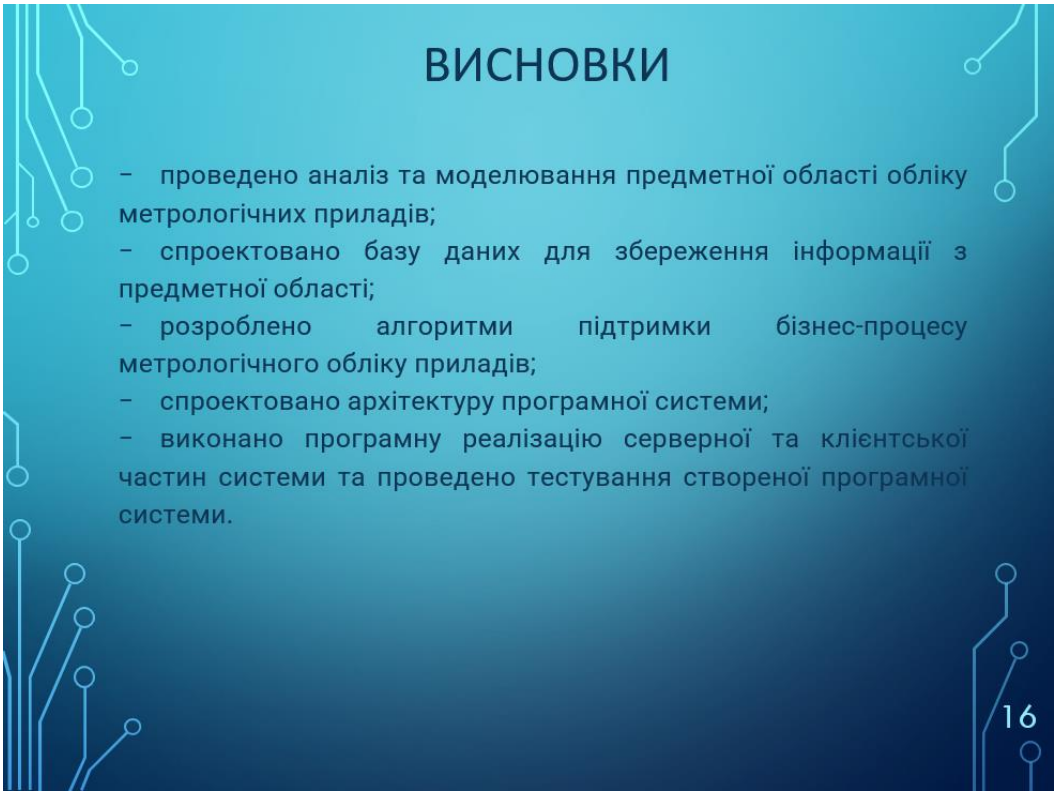


Рисунок Б.9 – Слайд «Висновки» (рисунок виконано самостійно)

ДОДАТОК В

Лістинг програмного коду

```
        private void customerToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            GlobalVar.Gl_FlagCustOrder = false;
            CustForm cs = new CustForm(this);
            cs.ShowDialog();
        }

        private void dataGridView_Orders_RowHeaderMouseClick(object
sender, DataGridViewCellMouseEventArgs e)
        {
            if (e.RowIndex < 0) return;
            DataGridViewRow selectedRow =
dataGridView_Orders.Rows[e.RowIndex];
            Ord_id = selectedRow.Cells[0].Value.ToString();
            GlobalVar.Gl_Ord_id = Ord_id;
            textBox_OrdNumber.Text =
selectedRow.Cells[1].Value.ToString();
            comboBox_OrdStatus.Text =
selectedRow.Cells[2].Value.ToString();
            textBox_OrdCust.Text = selectedRow.Cells[3].Value.ToString();
            textBox_OrdLetNum.Text =
selectedRow.Cells[4].Value.ToString();

            if (selectedRow.Cells[5].Value.ToString() != "")
            {
                dateTimePicker_Let_Date.Value =
Convert.ToDateTime(selectedRow.Cells[5].Value.ToString());
            }
            else dateTimePicker_Let_Date.Value = DateTime.Today;

            textBox_OrdCost.Text = selectedRow.Cells[6].Value.ToString();
            textBox_OrdVAT.Text = selectedRow.Cells[7].Value.ToString();

            if (selectedRow.Cells[8].Value.ToString() != "")
            {
                dateTimePicker_Pay.Value =
Convert.ToDateTime(selectedRow.Cells[8].Value.ToString());
            }
            else dateTimePicker_Pay.Value = DateTime.Today;

            if (selectedRow.Cells[9].Value.ToString() != "")
            {
                dateTimePicker_Rec.Value =
Convert.ToDateTime(selectedRow.Cells[9].Value.ToString());
            }
            else
                dateTimePicker_Rec.Value = DateTime.Today;

                textBox_OrdExec.Text =
selectedRow.Cells[10].Value.ToString();
```

```
        }

        private void button_SelCust_Click(object sender, EventArgs e)
        {
            GlobalVar.Gl_FlagCustOrder = true;
            CustForm cs = new CustForm(this);
            cs.ShowDialog();
        }

        private void button_SelExec_Click(object sender, EventArgs e)
        {
            SelectExec se = new SelectExec(this);
            se.ShowDialog();
        }

        private void button_OrdAdd_Click(object sender, EventArgs e)
        {
            if (textBox_OrdNumber.Text != "")
            {
                MessageBox.Show($"Треба очистити дані, \nта ввести нові
дані \nдля додавання!", "Додати",
                                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }
            if (comboBox_OrdStatus.Text == "" ||
                textBox_OrdCust.Text == "" ||
                textBox_OrdLetNum.Text == "")
            {
                MessageBox.Show("Треба ввести дані \nв усі поля з
зірочкою \nдля додавання!", "Додати",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }
            else
            {
                textBox_OrdNumber.Text = CreateOrdNum();
                SqlDataAdapter adapter = new SqlDataAdapter();
                DataTable table = new DataTable();
                DateTime dt =
DateTime.Parse(dateTimePicker_Let_Date.Text);
                string queryString = $"INSERT INTO Orders " +
                    $"(Ord_inv_num, Ord_status, Cust_id, " +
                    $"Ord_let_num, Ord_let_date, Ex_id) " +
                    $"VALUES ('{textBox_OrdNumber.Text}',
'{comboBox_OrdStatus.Text}', " +
                    $"'{GlobalVar.Gl_Cust_id}',
'{textBox_OrdLetNum.Text}', " +
                    $"DATEFROMPARTS({dt.Year},{dt.Month},{dt.Day}), " +
                    $"'{GlobalVar.Gl_Exec_id_OrdAdd}');";

                SqlCommand command = new SqlCommand(queryString,
db.getConnection());
                if (db.openConnection() > 0)
                {
                    adapter.SelectCommand = command;
                    adapter.Fill(table);
                    db.closeConnection();
```

```
                }
                else
                {
                    Application.Exit();
                }

                QueryOrders();
                //ClearCustTBox();
                return;
            }







        }

        private string CreateOrdNum()
        {
            string OrdNumber = "";
            string ONumMax = GetOrdNumMax();
            int ONum1;
            int ONum2 ;

            if (ONumMax == "" || ONumMax == null)
            {
                ONum2 = 0;
            }
            else
            {
                ONum2 = Convert.ToInt32(ONumMax)%10000;
            }

            ONum1 = Convert.ToInt32(GetDepNum()) * 10000;
            ONum1 = ONum1 + ONum2+1;
            OrdNumber = ONum1.ToString();
        return OrdNumber;
        }
        public string GetDepId()
        {
            string DepId = "";

            string qStr = $"SELECT Dep_id FROM Executor " +
                          $"WHERE Ex_id = '{GlobalVar.Gl_Exec_id}'";

            SqlCommand command = new SqlCommand(qStr,
db.getConnection());

            if (db.openConnection() > 0)
            {
                DepId = command.ExecuteScalar().ToString();
            }
            else
            {
                Application.Exit();
            }
            return DepId;
```

```
        }
        public string GetDepNum()
        {
            string DepNumber = "";

            string qStr = $"SELECT Dep_nums FROM Department " +
                          $"WHERE Dep_id = '{GlobalVar.Gl_Dep_id}'";

            SqlCommand command = new SqlCommand(qStr,
db.getConnection());

            if (db.openConnection() > 0)
            {
                DepNumber = command.ExecuteScalar().ToString();
            }
            else
            {
                Application.Exit();
            }
            return DepNumber;
        }
        public string GetOrdNumMax()
        {
            string OrdNumMax = "";

            string qStr = $"SELECT Max(Ord_inv_num) FROM Orders " +
                $"WHERE Ord_inv_num LIKE '{GetDepNum()}%'; ";


            SqlCommand command = new SqlCommand(qStr,
db.getConnection());

            if (db.openConnection() > 0)
            {
                OrdNumMax = command.ExecuteScalar().ToString();
                if (OrdNumMax == null) OrdNumMax = "00000";
            }
            else
            {
                Application.Exit();
            }
            return OrdNumMax;
        }
```