МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Центр післядипломної освіти
Кафедра програмної інженерії

КВАЛІФАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти: перший (бакалаврський)

Програмна система обліку робіт проектної організації
(тема)

Виконав:
Студент 4 курсу, групи ПЗПІп-22-1
Мішурова М. Г.
(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: к.т.н., доц. каф. ПЗ Вечур О. В.
(посада, прізвище, ініціали)

Допускається до захисту

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

Харків 2024

Харківський національний університет радіоелектроніки
Центр післядипломної освіти

Кафедра програмної інженерії

Рівень вищої освіти перший (бакалаврський)

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

Тип програми Освітньо-професійна

Освітня програма Програмна Інженерія
(шифр і назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мішуровій Марії Геннадіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система обліку робіт проектної організації

Затверджена наказом по університету від 17.06. 2024 р. № 588

2. Термін подання студентом роботи до екзаменаційної комісії 23.07.2024

3. Вихідні дані 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	Примітки
1	Аналіз проблемної області	07.05.2024 – 09.05.2024	виконано
2	Постановка задачі	10.05.2024 – 11.05.2024	виконано
3	Аналіз предметної галузі	12.05.2024 – 13.05.2024	виконано
4	Проектування бази даних	14.05.2024 – 21.05.2024	виконано
5	Проектування архітектури ПС	22.05.2024 – 27.05.2024	виконано
6	Розробка ПЗ	28.05.2024 – 11.07.2024	виконано
7	Підготовка пояснювальної записки	11.07.2024 – 15.07.2024	виконано
8	Нормоконтроль	16.07.2024	виконано
9	Рецензування	17.07.2024	виконано
10	Занесення диплома в архів	18.07.2024	виконано
11	Попередній захист	18.07.2024	виконано
12	Допуск до захисту у зав. кафедри	19.07.2024	виконано

Дата видачі завдання 7 травня 2024 р.

Студент (ка) _____ Мішурова М. Г.
(підпис)

Керівник роботи _____ к.т.н., доц. каф. ПІ Вечур О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Звіт до кваліфікаційної роботи містить: 58 с., 16 рисунків та 12 джерел.

Метою роботи є розробка десктопного додатку «Програмна система обліку робіт проектної організації» з використанням реляційних баз даних та логіки об’єктно-орієнтованого програмування.

В ході створення додатку були використані середа розробки Microsoft Visual Studio 2019, платформа ASP.Net, мова програмування C#, інтерфейс програмування додатків Windows Forms та система управління базами даних Microsoft SQL Server.

В результаті роботи був створений десктопний додаток, який дозволяє проводити облік Замовників, контрактів, проектів, співробітників та відділів, використане обладнання, надає можливість відстежувати відсоток виконання робіт в проектній організації. Також програма дозволяє шукати та фільтрувати інформацію, отримувати статистики замовлень, статистики відсотку виконання контрактів та формувати звіти в автоматичному режимі.

МОВА ПРОГРАМУВАННЯ C#, РЕЛЯЦІЙНА БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, ADO.NET, MICROSOFT SQL SERVER, WINDOWS FORMS, ІНТЕГРАЦІЙНЕ ТЕСТУВАННЯ, ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ, ТЕСТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА.

The report for the qualification work contains: 58 pages, 16 figures and 12 sources.

The aim of the work is to develop a dextop application "Software system for accounting of work of a project organisation" using relational databases and object-oriented programming logic.

The application was developed using the Microsoft Visual Studio 2019 development environment, ASP.Net platform, C# programming language, Windows Forms application programming interface, and Microsoft SQL Server database management system.

As a result of the work, a desktop application was created that allows you to keep track of customers, contracts, projects, employees and departments, equipment used, and provides an opportunity to track the percentage of work performed in the project organisation. The application also allows you to search and filter information, receive order statistics, contract completion statistics and generate reports automatically.

PROGRAMMING LANGUAGE: C#, RELATIONAL DATABASE, DATABASE MANAGEMENT SYSTEM, ADO.NET, MICROSOFT SQL SERVER, WINDOWS FORMS, INTEGRATION TESTING, FUNCTIONAL TESTING, USER INTERFACE TESTING.

Я, Мішурова М. Г., студентка гр. ПЗПІп-22-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система обліку робіт проектної організації», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

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

ЗМІСТ

Перелік скорочень 7
Вступ 8
1. Аналіз предметної галузі 10
1.1 Аналіз предметної галузі 10
1.2 Аналіз існуючих аналогів 11
1.3 Постановка задачі 13
1.4 Цільові користувачі 15
2. Формування вимог до програмної системи 16
2.1 Функціональні вимоги 16
2.2 Нефункціональні вимоги 17
2.3 Вимоги до звітності та аналітики 17
2.4 Системні обмеження 17
3. Архітектура та проектування програмного забезпечення 20
3.1 Загальні відомості 20
3.2 Побудова моделі даних 21
3.3 Функціональна структура проекту 22
3.4 Розгортання та запуск системи 26
3.5 Опис прийнятих програмних рішень 27
3.6 Логічна структура програми та приклади деяких методів 32
4. Тестування програмного забезпечення 36
4.1 Інтеграційне тестування 36
4.2 Функціональне тестування 37
4.3 Тестування інтерфейсу користувача 38
Висновки 39
Перелік джерел посилання 40
Додаток А Перевірка роботи на унікальність тексту 42
Додаток Б Презентація ПЗ 43
Додаток В Текстовий вивід програмного коду лістинг 49

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ПЗ – програмне забезпечення;

СКБД – система керування базою даних.

ВСТУП

Сучасний розвиток інформаційних технологій призвело до необхідності ефективного обліку та управління замовленнями та ресурсами у різних галузях, включаючи сферу проектної організації. Бізнес-середовище, що постійно змінюється, зростання конкуренції та вимоги до ефективності роботи підприємств припускають наявність сучасних та зручних інформаційних систем, здатних автоматизувати та покращити процеси обліку та контролю робіт.

Мета роботи полягає у розробці програмної системи, яка надасть організації можливість ефективно проводити облік та контролювати виконання проектів, а також покращити процеси обліку робіт, планування та аналізу проектних завдань, та загалом проводити облік всіх необхідних сфер. Створення такої системи має важливе практичне значення для підвищення продуктивності та оптимізації бюджетування в організації.

У результаті виконаної роботи було створено програмний продукт, що спрощує ведення обліку клієнтів, контрактів, проектів, співробітників та відділень у проектній організації, а також використаного обладнання. Під час виконання роботи було виконано концептуальне та логічне моделювання бази даних, створено ER-діаграму та реалізовано десктопну програму.

Програмний додаток є інтуїтивно зрозумілим і використовується керівниками, бухгалтерією, відділом кадрів та іншими співробітниками організації.

В ході створення додатку були використані середа розробки Microsoft Visual Studio 2019, платформа ASP.Net, мова програмування C#, інтерфейс програмування додатків Windows Forms [2] та система управління базами даних Microsoft SQL Server [3] безпосередньо для створення БД.

Перевірка роботи на унікальність тексту в мережі інтернет та базі ХНУРЕ (див. додаток А) показала відповідність вимогам державних стандартів з оформлення. Також, за результатами кваліфікаційної роботи створено презентацію (див. додаток Б) і розміщено текстовий вивід програмного коду лістинг (див. додаток В).

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної області

Об'єктом дослідження роботи є процес обліку робіт у проектній організації. Предметом дослідження є застосування сучасних технологій розробки програмних систем для покращення обліку робіт та планування проектів.

Практична цінність розробленого додатку для «Програмної системи обліку робіт проектної організації" полягає у безлічі переваг, які вона принесе проектній організації, а також, при невеликій переробці, і для інших організацій та підприємств суміжних галузей.

По-перше, дана програма дозволяє проектній організації ефективно вести облік та відстеження робіт. Надає можливість реєструвати та відстежувати інформацію про багато аспектів, таких як Замовники, виконані та актуальні роботи, витрачені ресурси, співробітників, використані матеріали у часових рамках. Точний облік робіт допоможе встановити прогрес проектів, виявити потенційні затримки та проблеми, а також своєчасно вживати заходів для їх вирішення.

По-друге, програма забезпечить можливість планування та контролю проектних завдань. Вона дозволить визначити завдання, необхідні досягнення цілей проектів, розподілити їх між виконавцями і встановити терміни виконання, надаватиме можливість відстежувати прогрес виконання завдань, контролювати їх статуси та своєчасно реагувати на зміни та зриви графіка.

По-третє, програма надає інструменти для управління ресурсами проекту, дозволяє оптимізувати розподіл людських та матеріальних ресурсів, а також встановити ефективну систему контролю за їх використанням. Також сповіщає про необхідність перекваліфікації співробітників та повірки приладів, коли спливає термін їх сертифікатів.

По-четверте, розроблений програмний додаток дозволяє створювати звіти, які слугуватимуть важливим інструментом для прийняття рішень та

оцінки ефективності проектів. Звіти включають інформацію про Замовників та співробітників, буде додано звіти інформацією про витрати, виконання робіт, використання ресурсів та інші показники проектів. Аналітика допоможе виявити тренди, аномалії та прогнозувати можливі ризики, що зробить управління проектами більш обґрунтованим та раціональним.

У результаті, розробляємий програмний продукт представляє наочну інформацію про найважливіші аспекти організації, покращує управління проектами та підвищує продуктивність проектної організації, що значно оптимізує витрати. Завдяки впровадженню даної системи, організація зможе більш ефективно планувати, контролювати та керувати своїми проектами та ресурсами, що призведе до підвищення її конкурентоспроможності та успіху на ринку.

1.2 Аналіз існуючих аналогів

Проведено аналіз ринку, у ході якого з’ясовано, що існує деякі програмні продукти, що мають дещо схожий функціонал до «Програмної системи Облік робіт проектної організації», можна розглянути подібні платформи та інструменти, що спрямовані на управління проектами та відстежування робіт. Нижче подані деякі можливі аналоги:

– Jira [6] від Atlassian, це платформа для управління проектами та відстеженням робіт, яка широко використовується в IT-проектах. Перевагами виступають велика гнучкість у конфігурації завдань, велика спільнота користувачів, інтеграція з іншими продуктами Atlassian, а недоліками виступають доволі таки складний інтерфейс (див. рис. 1.1) через великий обсяг можливостей, який може бути і не всім потрібний, немає україномовного інтерфейсу, а також чуттєва вартість користування;

– Asana [7], це сервіс для організації робочих завдань та управління проектами, перевагами виступають легкий у використанні та інтуїтивний інтерфейс (див. рис. 1.2), наявність широких можливостей спільної роботи, а

недоліками виступають обмежена функціональність для складних проектів, відсутність україномовного інтерфейсу та платні додаткові можливості;

– Trello [8] від Atlassian, це інструмент для керування завданнями та проектами у формі дошок, перевагами є простий та зрозумілий інтерфейс (див. рис. 1.3), велика кількість додатків та інтеграцій, недоліками виступають обмежена функціональність для великих проектів, може виглядати занадто простим для деяких завдань.

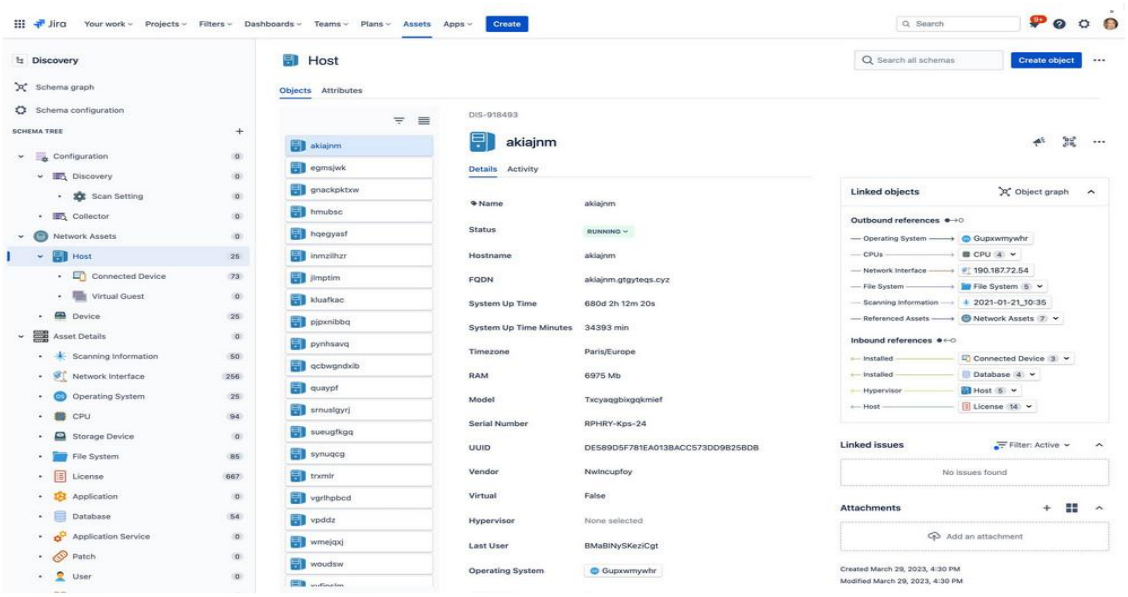


Рисунок 1.1 – Інтерфейс платформи для управління проектами «Jira» ([6])

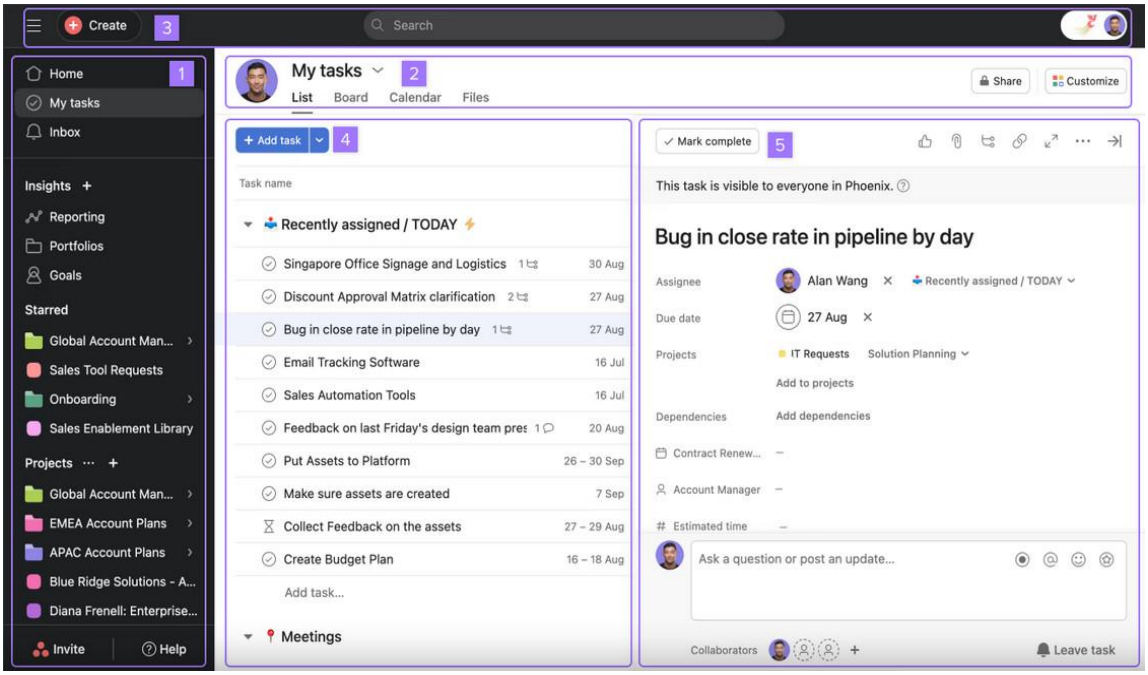


Рисунок 1.2 – Інтерфейс сервісу для управління проектами «Asana» ([7])

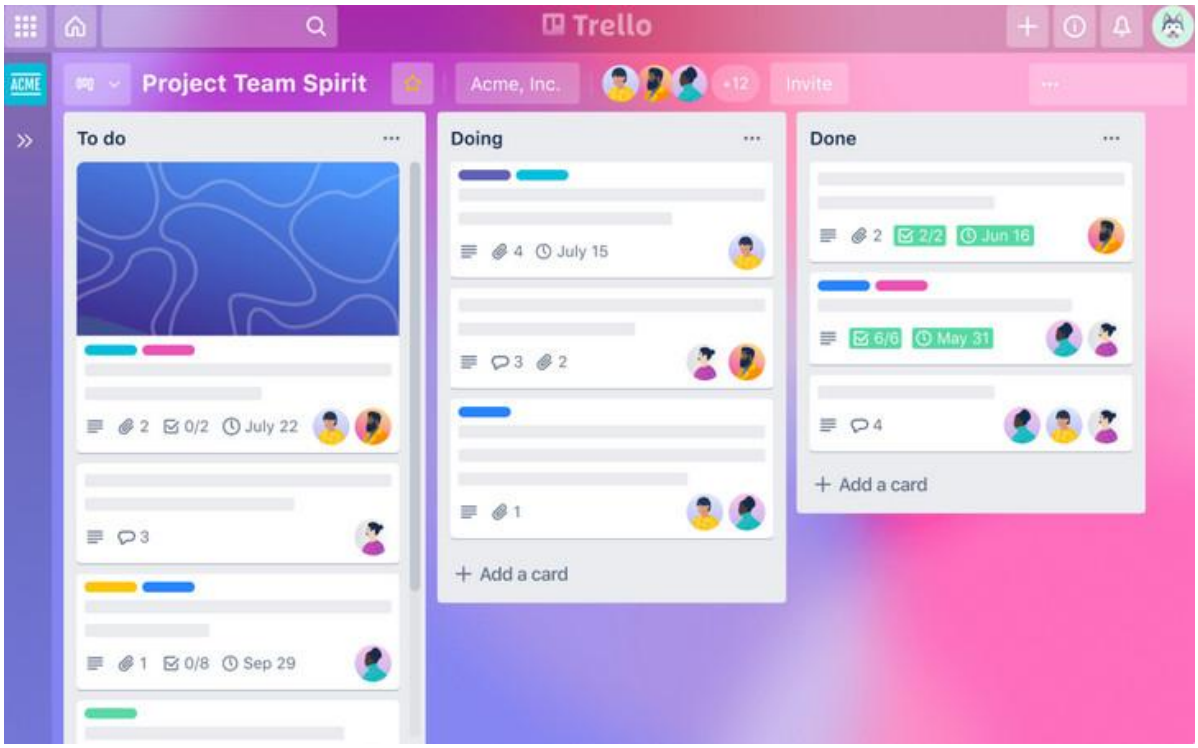


Рисунок 1.3 – Інтерфейс інструменту для керування завданнями «Trello» ([7])

1.3 Постановка задачі

Спочатку було розроблено базу даних для збереження і керування інформацією, пов'язаною з процесами та даними в проектній організації, а також створено програмну систему, яка виконує наступні завдання:

– відображає усі данні безпосередньо про основні поняття проектної організації: Замовники і контракти, співробітники, проекти та документація, прилади та сертифікати до них;

– відображає пов’язані поняття ПЗ: відділи, види робіт, використане обладнання та свідотство до них;

– система підтримує арифметичну обробку даних у вигляді обчислювальних полів стосовно загальної кількості контрактів, загальної вартості договорів, полів стосовно строків виконання робіт по договорам та робить підрахунок відсотку виконання контрактів на основі записаних виконавцями відсотків;

– система підтримує сортування, пошук та фільтрацію даних:

а) виконується пошук Замовника за його назвою чи кодом ЄДРПОУ, виконується пошук контракту також за назвою чи за його шифром;

б) сортування контрактів за замовниками та за статусом, сортування проектів за контрактами, а співробітників – за відділами;

в) здійснюється фільтрація Замовників за кількістю замовлень, а контрактів – за періодом виконання та статусом;

– система підтримує додавання нових даних, з підтримкою режиму підтвердження, про Замовників, співробітників, контрактів, проектів, видів робіт, використаного обладнання та інших супутніх даних;

– система підтримує можливість редагування інформації, з підтримкою режиму підтвердження, про Замовників, співробітників, контрактів, проектів, використаного обладнання та інших супутніх даних;

– система підтримує можливості вилучення інформації про Замовників, співробітників, контрактів, проектів, використаного обладнання та інших супутніх даних з підтримкою режиму підтвердження користувачем видалення інформації про поточний об’єкт;

– система підтримує виконання наступних часто виникаючих запитів до бази даних:

а) статистику рейтингів Замовників за кількістю контрактів;

б) статистику контрактів за їх вартістю;

в) статистику бізнес розвитку компанії у часі (кількість виконаних контрактів за рік);

г) статистику за відсотком виконання контрактів.

– система повинна підтримувати можливість формування довільного запиту до БД на мові SQL з підтримкою користувача інформацією стосовно схеми БД;

– система повинна підтримувати підготовку та друк наступних звітів:

а) звіт по співробітникам (ПІБ співробітника, посада, стаж, дата прийняття, сертифікати, оклад);

б) звіт по Замовникам (найменування Замовника, код ЄДРПОУ, розрахунковий рахунок, кількість замовлень, електронна пошта компанії, ПІБ контактної особи та телефон).

1.4 Цільові користувачі

Основними видами користувачів на даному етапі розробки додатку є:

- керівний склад організації, це безпосередньо директора, а також ГІП, головний спеціаліст, керівник групи;
- керівники підрозділів та співробітники організації;
- бухгалтерія та відділ кадрів.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Вимоги до програмної системи [9] – специфікації, які визначають, що повинна робити система та як вона повинна працювати. Вони є основою для розробки, тестування та впровадження програмного забезпечення.

Вимоги поділяються на дві основні категорії:

– функціональні вимоги, які визначають конкретні функції та завдання, які система повинна виконувати, описують поведінку системи у різних ситуаціях;

– нефункціональні вимоги, які описують, як система повинна виконувати свої функції, тобто якість її роботи, включають аспекти, такі як продуктивність, безпека, надійність, масштабованість, зручність користування та інші.

2.1 Функціональні вимоги

Облік робіт та ресурсів: програма повинна забезпечувати можливість реєстрації, обліку та відстеження виконаних робіт, використаних матеріалів, витрачених ресурсів, а також інформації про замовників та співробітників. Важливою функцією є ведення хронології виконання робіт та витрат ресурсів для кожного проекту.

Планування та контроль проектних завдань: програма повинна надавати інструменти для планування проектних завдань, включаючи визначення завдань, встановлення термінів їх виконання, контроль за виконанням завдань по співробітникам. Повинна бути можливість відстежувати статус виконання завдань та реагувати на можливі зміни.

Управління ресурсами: система повинна надавати функціонал для оптимізації розподілу людських та матеріальних ресурсів, контролю за їх використанням, а також своєчасного сповіщення про необхідність повірки приладів, коли спливає термін їх сертифікатів.

2.2 Нефункціональні вимоги

Продуктивність: програма повинна забезпечувати високу продуктивність та швидкість обробки даних для забезпечення безперебійної роботи при великій кількості одночасних користувачів та значному обсязі даних.

Надійність та безпека: система повинна гарантувати збереження та цілісність даних, забезпечуючи резервне копіювання та відновлення даних у разі збоїв. Важливо також забезпечити захист конфіденційної інформації за допомогою сучасних методів аутентифікації та авторизації користувачів.

Юзабіліті: інтерфейс програми повинен бути інтуїтивно зрозумілим та зручним для користувачів різного рівня кваліфікації. Навігація повинна бути простою, а функціонал доступним та легко освоюваним.

2.3 Вимоги до звітності та аналітики

Звіти: повинна надавати можливість створення різноманітних звітів, що включають інформацію про замовників, співробітників, витрати, виконання робіт та використання ресурсів. Звіти повинні бути налаштованими для різних потреб користувачів.

Аналітика: програма повинна забезпечувати інструменти для аналізу даних, виявлення трендів, аномалій та прогнозування можливих ризиків. Це дозволить підвищити обґрунтованість управлінських рішень та ефективність проектного менеджменту.

2.4 Системні обмеження

Програмна система обліку робіт проектної організації має задовольняти наступним вимогам щодо інтерфейсу:

– стандартний інтерфейс: ПС повинна мати стандартний інтерфейс, який відповідає прийнятому стилю і дизайну програмних програм для

операційної системи MS Windows. Це забезпечить зручність використання та інтуїтивно зрозумілу роботу з системою для користувачів.

– ергономіка: інтерфейс ІС повинен бути ергономічним і зручним для користувачів. Метою є мінімізація складності та зусиль, необхідних для роботи з системою.

– навігація: інтерфейс повинен забезпечувати зручну навігацію за функціями і розділами ІВ. Наприклад, можна включати меню, панелі інструментів, вкладки та інші елементи, які допоможуть користувачам швидко переходити до потрібних розділів і виконувати необхідні дії.

Вимоги до продуктивності: Особливих вимог до продуктивності програмної системи немає.

Для забезпечення безпеки Програмної системи обліку робіт проектної організації мають бути реалізовані такі вимоги:

– автентифікація користувачів: користувачі повинні входити в систему під своїми унікальними іменами і захищеними паролями. Це забезпечить ідентифікацію і запобігатиме несанкціонованому доступу до системи.

– керування доступом: Доступ до різних функцій і даних в програмі повинен бути обмежений відповідно до прав користувачів. Адміністратор користувачів повинен мати можливість додавати, змінювати і видаляти користувачів, а також призначати їм відповідні права доступу.

– конфіденційність даних: програмна система повинна забезпечувати конфіденційність збережених даних (система має виконувати Закон України «Про захист персональних даних» от 01.06.2010 № 2297-VI). Це може бути досягнуто шляхом шифрування даних.

– резервне копіювання і відновлення: програмна система повинна регулярно виконувати резервне копіювання даних для запобігання втрати інформації.

Експлуатаційні вимоги: програмна система має функціонувати на ОС Windows 8, ОС Windows 10, ОС Windows 11. Мінімальні апаратні вимоги визначаються мінімальними апаратними вимогами до перерахованих вище ОС.

Інших обмежень немає.

3 АРХІТЕКТУРА ПРОЕКТУВАННЯ

3.1 Загальні відомості

Для втілення даного додатку було використано середу розробки Microsoft Visual Studio 2019 та мову програмування C#. Для створення візуального представлення програми було обрано інтерфейс програмування додатків Windows Forms, що є частиною Microsoft .NET Framework [2]. Даний інтерфейс дозволяє швидко створити зручний та інтуїтивно зрозумілий додаток.

Додаток націлено на взаємодію із базами даних, отже для їх створення було обрано систему управління реляційними базами даних Microsoft SQL Server [3, 5]. Саме ця СУБД була використана через її здатність швидкої обробки інформації, легкість використання та інтегрованість з іншими продуктами Microsoft. За допомогою утиліти SQL Server Management [5] Studio було здійснено управління та конфігурування компонентів Microsoft SQL Server.

Доступ до баз даних було здійснено за технологією ADO.NET[4]. 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 веб-застосунків 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 я і надала перевагу ADO.NET, де з'єднання встановлюється лише на той короткий час, коли необхідно проводити операції над базою даних. Але ця технологія не буде доречною при використанні програм, які проводять часті та об'ємні зміни змісту записів.

3.2 Побудова моделі даних

Діаграма сутностей - зв'язків (також відома як діаграма ERD або ER [10]) це тип блок-схеми, який показує, як різні «сутності» (люди, об'єкти, поняття тощо) взаємопов'язані всередині системи.

ER-діаграми найчастіше використовуються для проектування і налагодження реляційних баз даних в сфері освіти, досліджень і розробок програмного забезпечення та інформаційних систем для бізнесу. Ці діаграми розташовані за тим же принципом, що і граматичні конструкції: сутності виступають іменниками, а зв'язки - дієсловами.

Було розроблена ER-діаграма за нотацією Баркера (див. рис. 3.4).

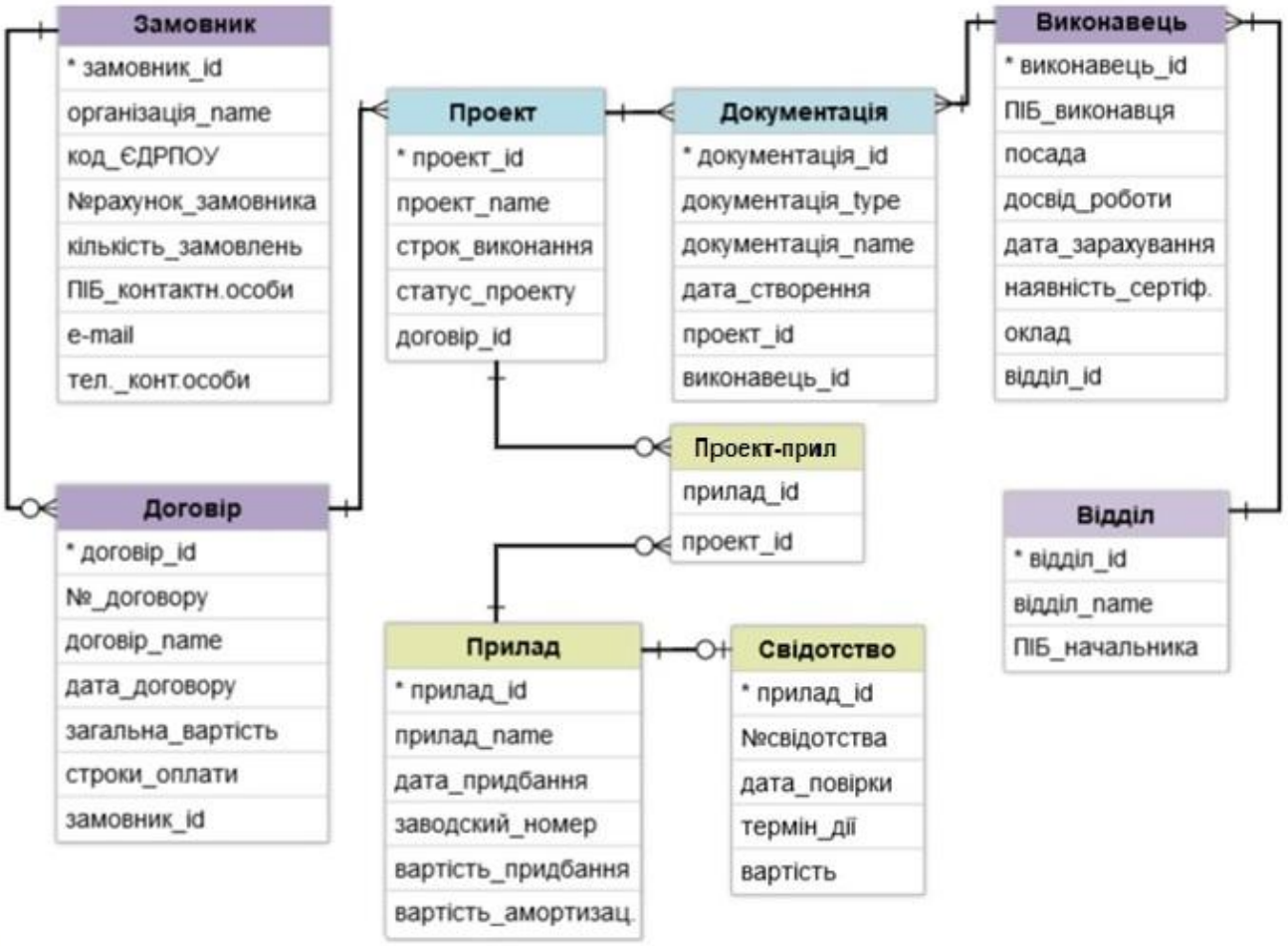


Рисунок 3.4 – ER-діаграма за нотацією Баркера (виконано самостійно)

Реляційна база даних – це база даних, яка використовується для зберігання та організації доступу до взаємопов’язаних елементів інформації. Реляційні бази даних ґрунтуються на реляційній моделі – 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. Саме цьому, реляційна модель бази даних є найбільш ефективною для вирішення задачі даного курсового проекту.

3.3 Функціональна структура проекту

Опис функціонального призначення програмної системи, що створюється, наведено на рисунку 3.5.

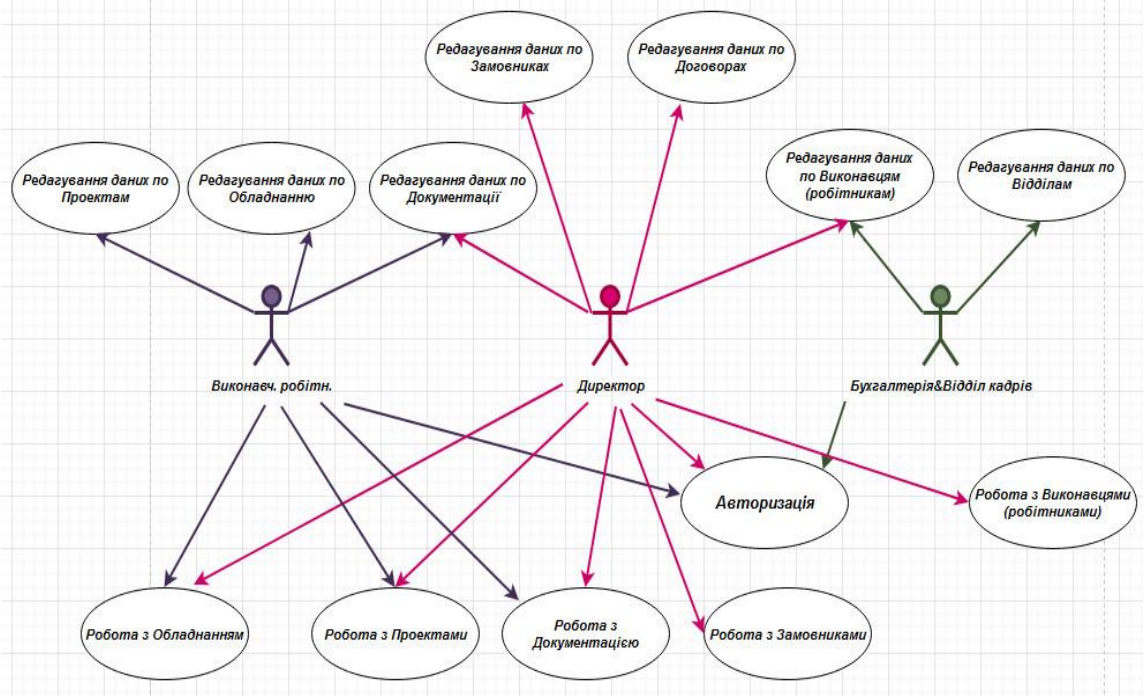


Рисунок 3.5 – Загальна USE-CASE діаграма (виконано самостійно)

Загальна діаграма класів на рисунку 6 відображає опис концептів програмної області, їх властивостей та зв'язків між ними.

Рисунок 3.6 – Загальна діаграма класів (виконано самостійно)

Розглянемо перелік характеристики для кожного з наведених понять. Поняття Замовник має наступні властивості: назва підприємства, код ЄДРПОУ, розрахунковий рахунок, кількість виконаних замовлень, електронна пошта підприємства, ПІБ контактної особи, телефон контактної особи. Поняття виконавець включає в себе ПІБ виконавця, його посада, дата прийняття, стаж, оклад, наявність сертифікатів. Контракти мають шифр, назву, дату підписання, дату завершення, вартість та статус завершення.

Для продуктивної роботи користувачів додатком, під час роботи з інформацією, вони мають такі можливість:

– додавати, редагувати та видаляти дані про Замовників та контракти;

– виконувати пошук Замовника за його назвою чи кодом ЄДРПОУ, виконувати пошук контракту також за назвою чи за його шифром;

– сортувати контракти за замовниками, за статусом;

– здійснювати фільтрацію Замовників за кількістю замовлень, а контрактів – за періодом виконання.

Користувачі мають можливість отримання наступних статистик для наявного та наглядного представлення інформації про організацію:

– статистику рейтингів Замовників за кількістю контрактів;

– статистику контрактів за їх вартістю;

– а також статистику бізнес розвитку компанії у часі (кількість виконаних контрактів за рік).

Опис існуючого документообігу складається з наступних документів:

– звіт по співробітникам (ПІБ співробітника, посада, стаж, дата прийняття, сертифікати, оклад);

– звіт по Замовникам (найменування Замовника, код ЄДРПОУ, розрахунковий рахунок, кількість замовлень, електронна пошта компанії, ПІБ контактної особи та телефон).

Також, в програмній системі для відповідальних користувачів програма виконує сповіщення про необхідність перекваліфікації співробітників та повірки приладів, коли спливає термін їх сертифікатів.

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

– кожен Замовник може заключити один чи багато контрактів;

– виконавець може одночасно брати участь в одному чи багатьох контрактів;

– виконавець може працювати над контрактами багатьох Замовників, а Замовники можуть відправляти багато контрактів.

З урахуванням ліцензійного програмного забезпечення необхідно дотримуватися наступних обмежень:

– всі документи формувати для редакторів MS Word або MS Excel;

– використовувати СУБД MS SQL Server.

Схематично зобразимо інформаційні потреби різних видів користувачів підприємства (див. рис. 3.7).

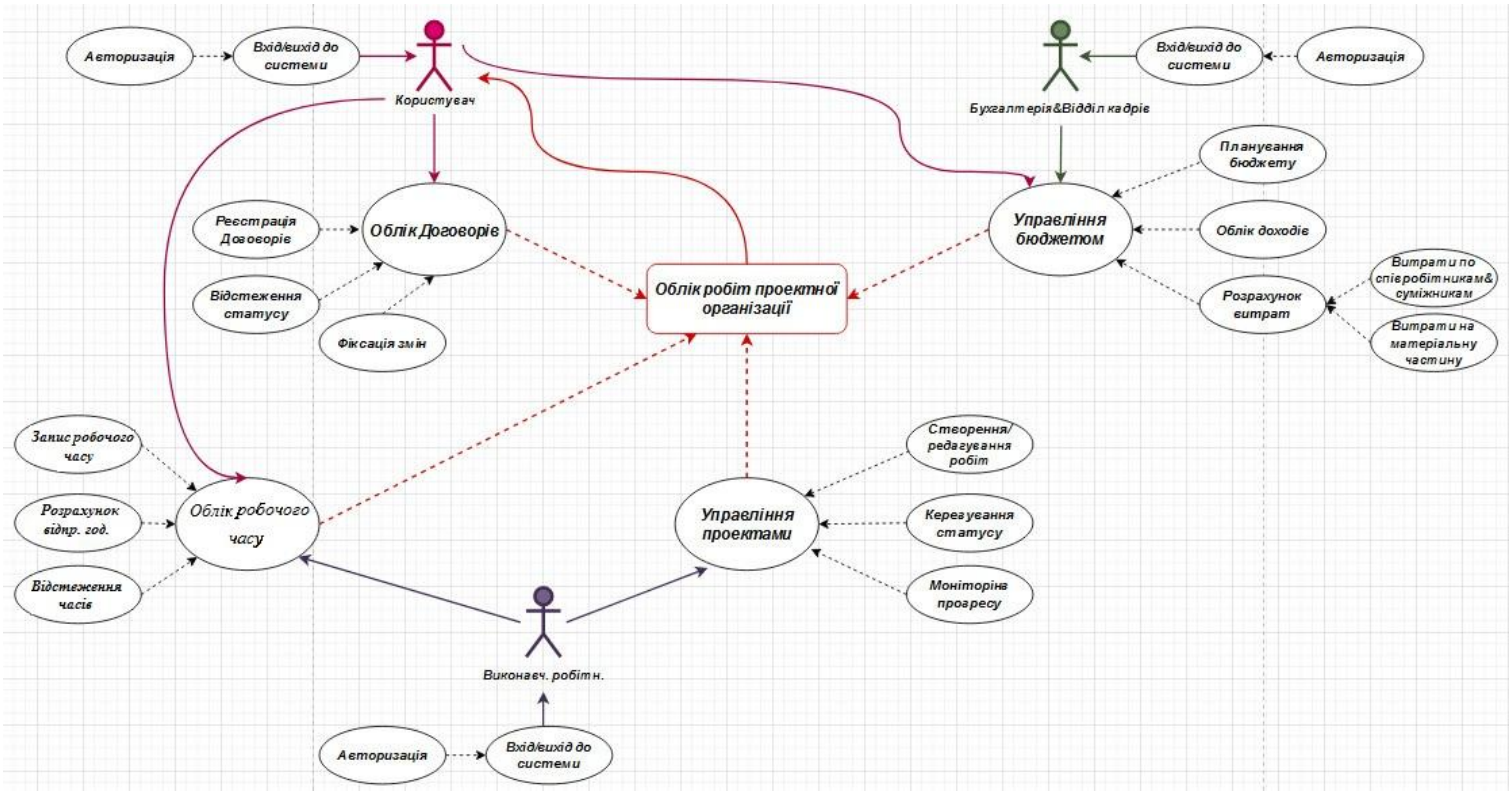


Рисунок 3.7 – Детальна USE-CASE діаграма (виконано самостійно)

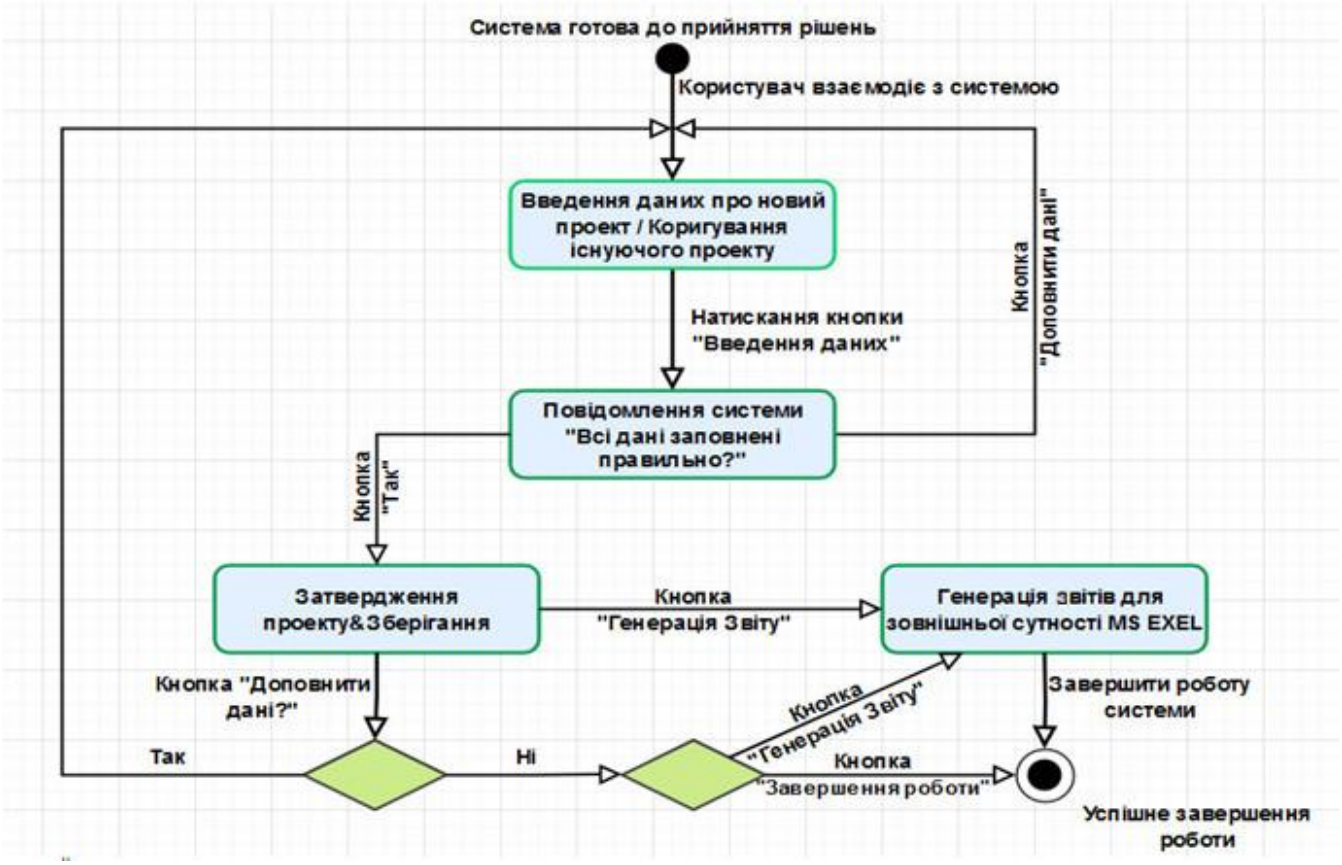


Рисунок 3.8 – Діаграма станів (виконано самостійно)

Для розуміння конкретного алгоритму дій користувача та реакції програмної системи на його дії у процесі проектування розроблено діаграму станів (див. рис. 3.8) для функції процесу управління проектами. Вона показує, як об'єкт переходить з одного стану в інший. Діаграми станів служать для моделювання динамічних аспектів системи.

3.4 Розгортання та запуск системи

Для запуску застосунку на комп’ютері користувач має завантажити відповідний файл з назвою Curs_DB.exe. Також, для роботи з додатком необхідно імпортувати базу даних. Для цього у користувача має бути встановлений SQL Server и SQL Server Management studio [3, 5]. Для імпорту необхідно виконати наступні дії:

– запустити SSMS та під ’єднатися до SQL сервера;

– в меню «Object Explorer» обрати гілку «Databases» та обрати пункт меню «Restore Datebase…»;

– у відкритому меню обрати опцію завантаження бази даних з ПК та перенести туди попередньо завантажену копію Project_DesignCompany.

Після успішного імпорту користувач може запускати файл Curs_DB.exe та починати роботу, а саме здійснити вхід в систему, використовуючи свої облікові дані (логін та пароль). Далі користувач отримує доступ до інтерфейсу системи (залежить від прав доступу) та взаємодіє з системою, використовуючи доступні функції та опції. Він може виконувати різні дії, такі як перегляд, створення, редагування та видалення записів в системі.

Ця інструкція надає загальний огляд функціоналу та основних кроків, необхідних для використання програмної системи.

3.5 Опис прийнятих програмних рішень

Після запуску програми з боку Керівництва відкривається вікно «Замовники», зовнішній вигляд якого наведено на рисунку 9.

Головне меню дозволяє переходити на додаткові вікна програми, а також формувати статистики та звіти. Користувач у цьому вікні може виконувати додавання, редагування та видалення Замовників, виконувати пошук за назвою організації Замовника та його ЄДРПОУ, виконувати фільтрацію за кількістю замовлень. В текст-боксах можна наглядно і повно продивитися інформацію по Замовнику, виділивши відповідну строчку в таблиці, таким же чином виконується і редагування та додавання. Програма, при будь-якій активності користувача, буде видавати попередження про зміни у додатку (приклад див. рис. 10).

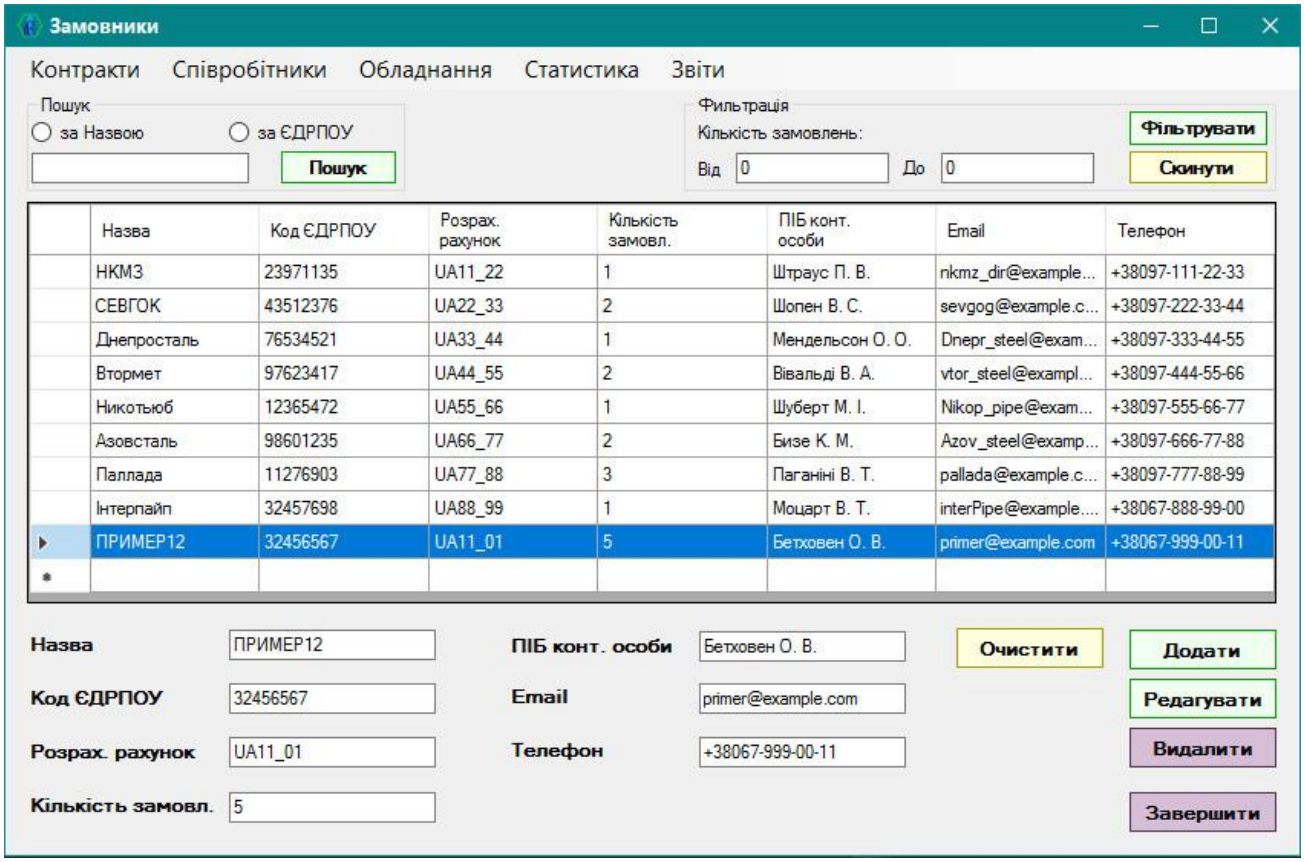


Рисунок 3.9 – Головне вікно програми, вікно «Замовники» (виконано самостійно)

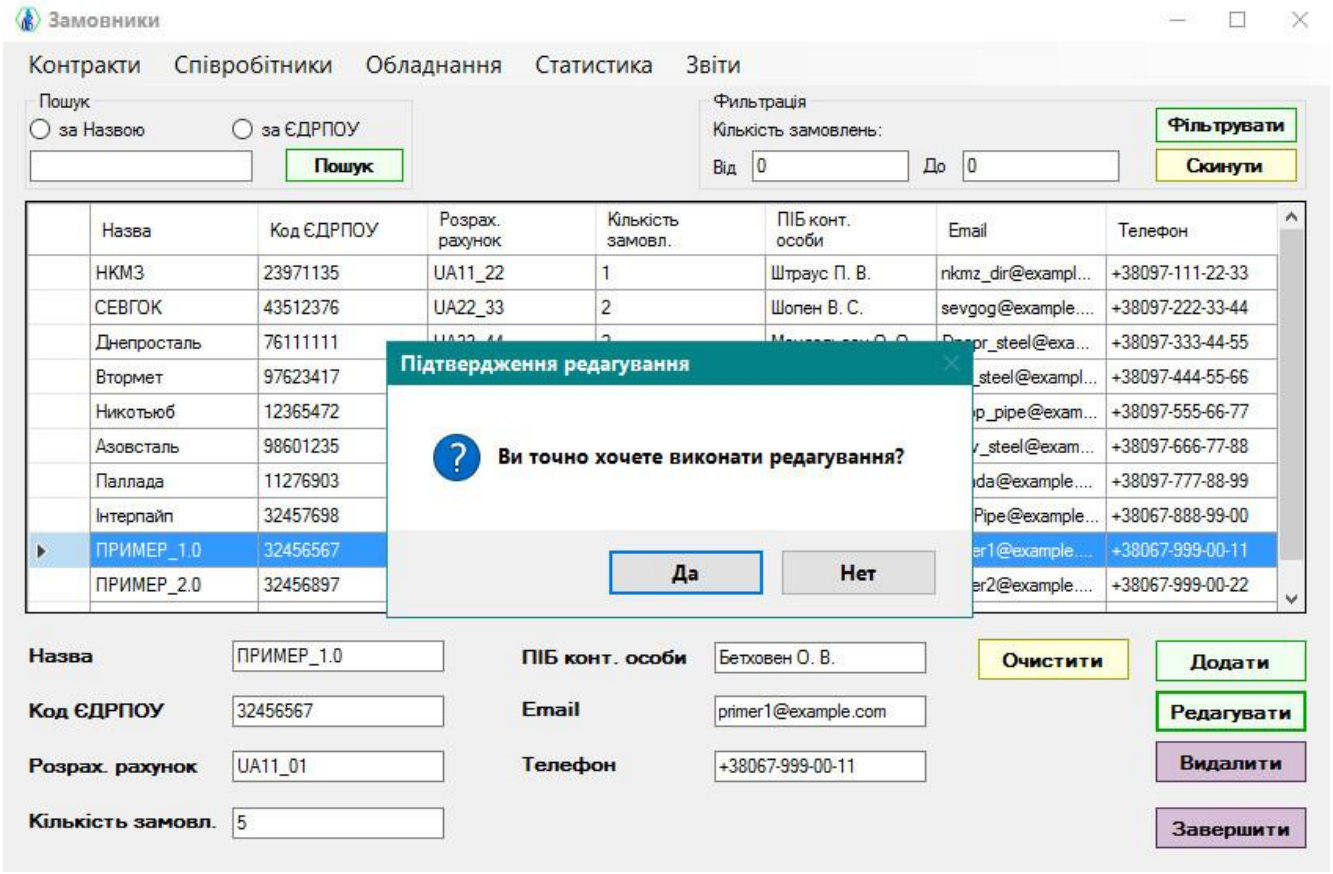


Рисунок 3.10 – Запит програми на дозвіл редагування (виконано самостійно)

У вікні програми «Контракти» також можна виконувати ті ж дії з додаванням, редагуванням та видаленням контрактів і можна з випадаючого списку (див. рис. 3.11) вибирати Замовників, контракти яких тоді будуть показані.

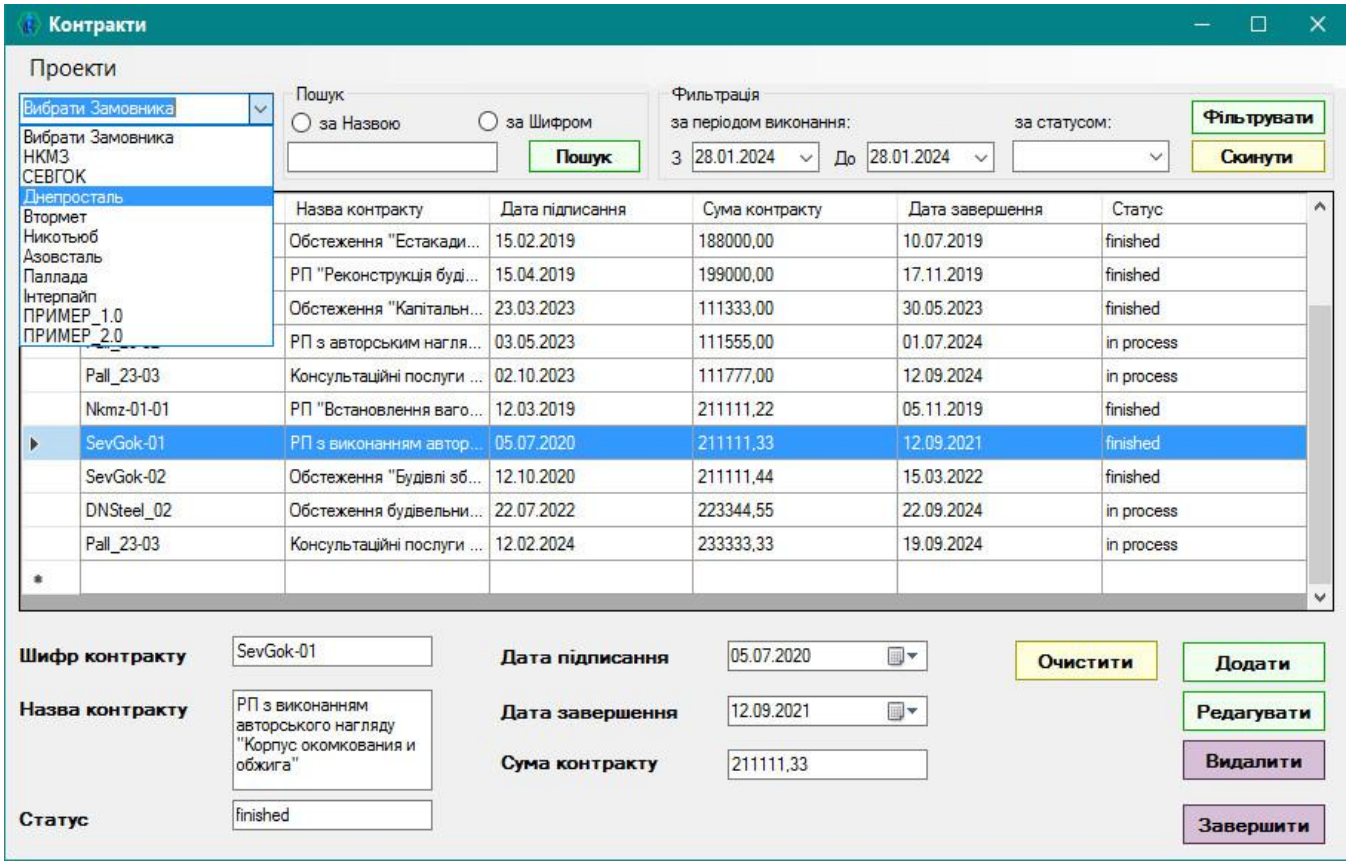


Рисунок 3.11 – Реалізація випадаючого списку у вікні «Контракти» (виконано самостійно)

Були представлені три статистики:

– рейтинг Замовників по кількості контрактів (див. рис. 3.12);

– рейтинг Контрактів за їх вартістю (див. рис. 3.13);

– бізнес розвиток компанії за кількістю Контрактів (див. рис. 3.14);

– Відсоток виконання контрактів.

В цих вікнах також були добавлені додаткові текст-бокси для спрощення перегляду необхідної інформації.

З меню в головному вікні можна обрати один з двох звітів – звіт по Співробітникам та по Замовникам. Скріншот наведено на рисунку 3.15 (вікно формування з кнопкою формування звіту) та сформовані звіти.

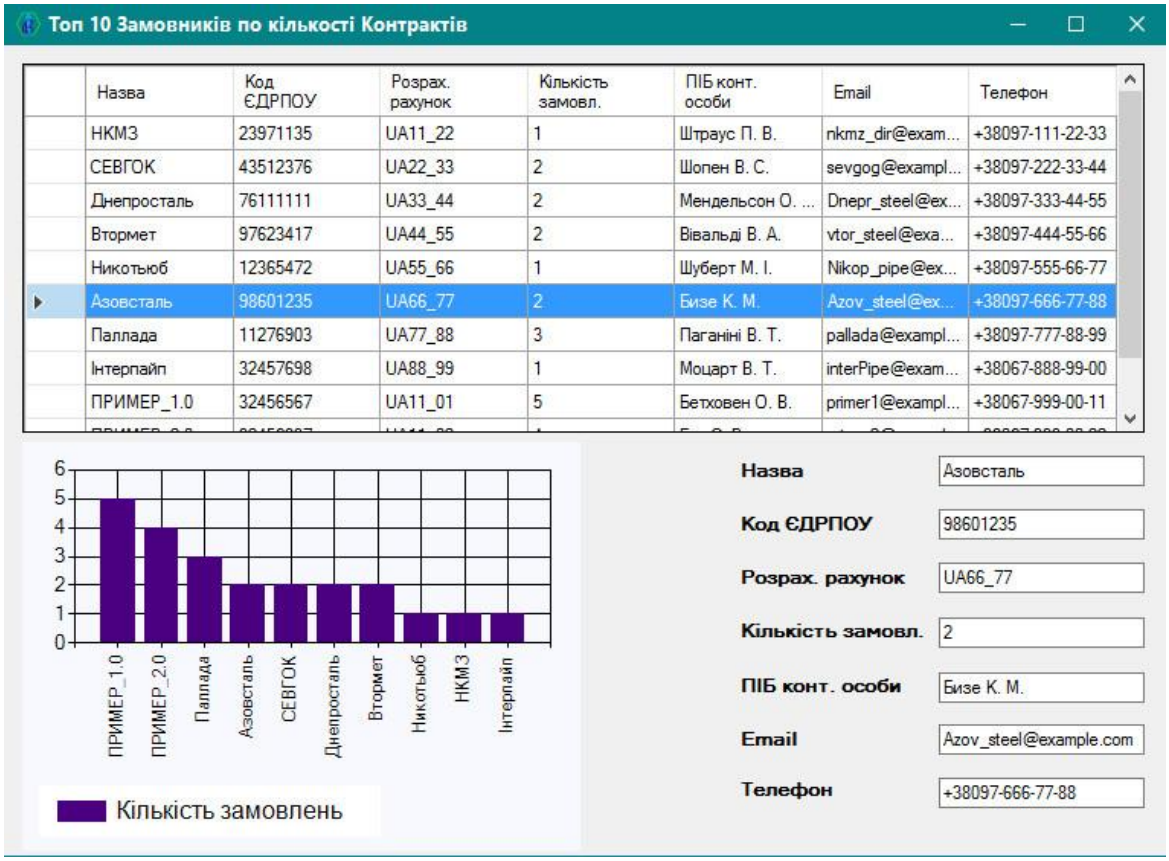


Рисунок 3.12 – рейтинг Замовників по кількості контрактів (виконано самостійно)

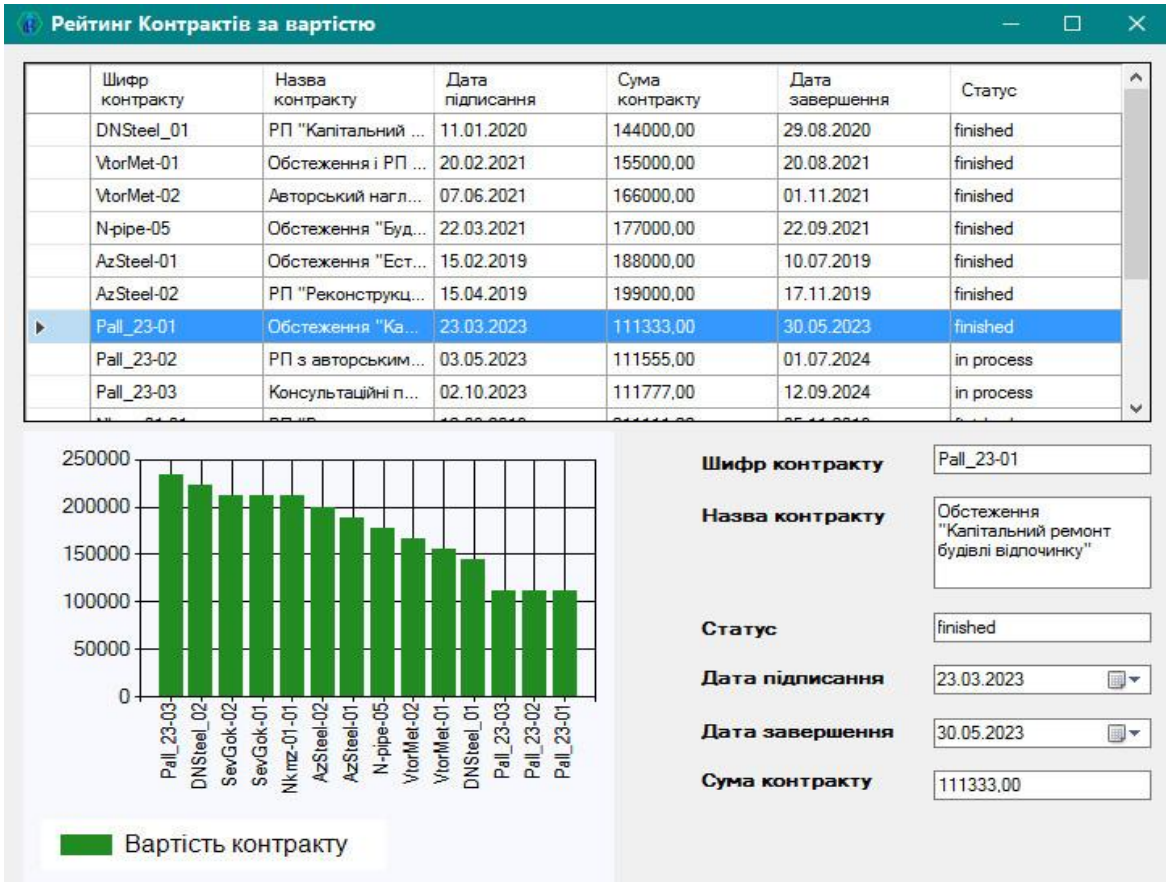


Рисунок 3.13 – рейтинг Контрактів за вартістю (виконано самостійно)

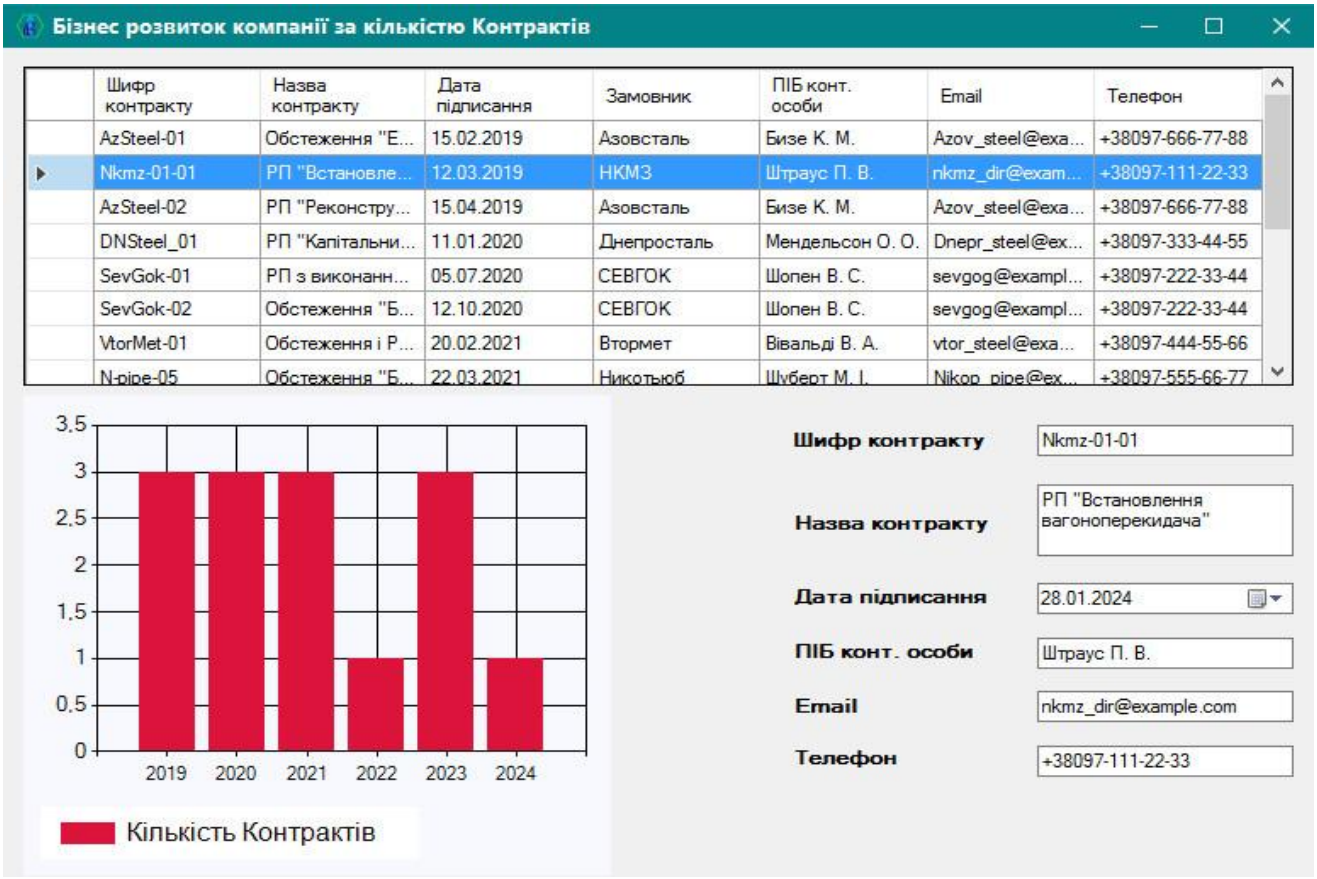


Рисунок 3.14 – Бізнес розвиток компанії за кількістю Контрактів (виконано самостійно)

Рисунок 3.15 – Звіт по Замовники, вікно формування (виконано самостійно)

В головному вікні меню після вибору звітів формуються самі звіти, які можна переглянути чи відправити на печать.

3.6 Логічна структура програми та приклади деяких методів

Логічна структура декстопного додатку приведена на рисунку 3.16

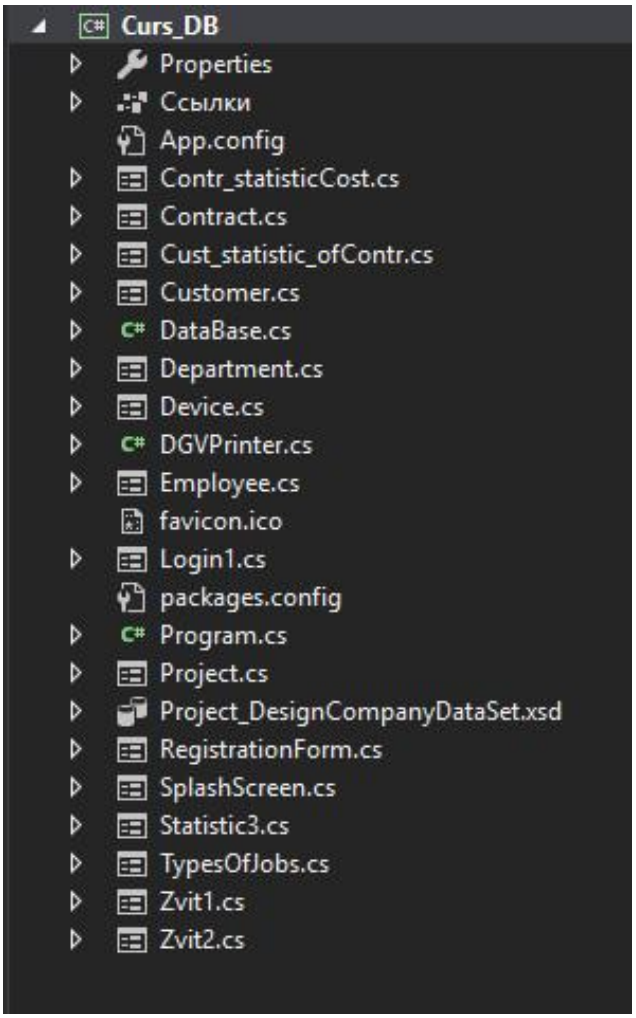


Рисунок 3.16 – Логічна структура програмної системи «Облік робіт проектної організації» (виконано самостійно)

Нижче представлено метод filter_Click, який реалізує функціональність фільтрації даних у таблиці "Замовники" за кількістю замовлень. Метод приймає в якості параметрів подію та об'єкт відправника події (зазвичай це кнопка, за кліком на яку викликається цей метод). Залежно від введених користувачем значень, метод формує SQL-запит, виконує його і відображає відфільтровані дані в таблиці на формі.

```
private void filter_Click(object sender, EventArgs e)
        {
            int start = Convert.ToInt32(textBox_start.Text);
            int finish = Convert.ToInt32(textBox_finish.Text);
```

```
            if (start <= 0 && finish <= 0)
            {
                QueryCust();
            }
            if ((start <= 0 && finish > 0) || (start > 0 && finish > 0))
            {
                SqlDataAdapter dataAdapter = new SqlDataAdapter();
                DataTable table = new DataTable();

                string query = $"SELECT Cust_id, Cust_organization AS
'Назва', Cust_code AS 'Код ЄДРПОУ', " +
                    $"Cust_AccauntNumber AS 'Розрах. рахунок',
Cust_Orders AS 'Кількість замовл.', " +
                    $"Cust_name AS 'ПІБ конт. особи', Cust_email AS
'Email', Cust_tel AS 'Телефон' FROM  Customer " +
                    $"WHERE Cust_Orders BETWEEN {start} AND {finish} ";

                SqlCommand command = new SqlCommand(query,
db.getConnection());
                if (db.openConnection() > 0)
                {
                    dataAdapter.SelectCommand = command;
                    dataAdapter.Fill(table);
                    dataGridView_cust.DataSource = table;
                    dataGridView_cust.Columns[0].Visible = false;
                    db.closeConnection();
                }
            }
            if (start > 0 && finish <= 0)
            {
                int newFinish = 100000;

                SqlDataAdapter dataAdapter = new SqlDataAdapter();
                DataTable table = new DataTable();

                string query = $"SELECT Cust_id, Cust_organization AS
'Назва', Cust_code AS 'Код ЄДРПОУ', " +
                    $"Cust_AccauntNumber AS 'Розрах. рахунок',
Cust_Orders AS 'Кількість замовл.', " +
                    $"Cust_name AS 'ПІБ конт. особи', Cust_email AS
'Email', Cust_tel AS 'Телефон' FROM  Customer " +
                    $"WHERE Cust_Orders BETWEEN {start} AND {newFinish}
";

                SqlCommand command = new SqlCommand(query,
db.getConnection());
                if (db.openConnection() > 0)
                {
                    dataAdapter.SelectCommand = command;
                    dataAdapter.Fill(table);
                    dataGridView_cust.DataSource = table;
                    dataGridView_cust.Columns[0].Visible = false;
                    db.closeConnection();
                }
            }
              }
```

Нижче наведено код реалізації методу для пошуку. Цей код дозволяє користувачеві швидко знайти необхідного замовника в базі даних за назвою організації або за кодом ЄДРПОУ. Він працює так само, як пошук в Google: вводимо ключові слова, і система знаходить всі записи, які відповідають необхідному запиту.

```
private void button_Search_Click(object sender, EventArgs e)
        {
            string queryString = "";

            if (textBox_Search.Text != "")
            {
                if (radioButton_Name.Checked)
                {
                    queryString = $"SELECT Cust_id, " +
                                  $"Cust_organization AS 'Назва', " +
                                  $"Cust_code AS 'Код ЄДРПОУ', " +
                                  $"Cust_AccauntNumber AS 'Розрах.
рахунок', " +
                                  $"Cust_Orders AS 'Кількість замовл.', "
+
                                  $"Cust_name AS 'ПІБ конт. особи', " +
                                  $"Cust_email AS 'Email', " +
                                  $"Cust_tel AS 'Телефон' " +
                                  $"FROM Customer " +
                                  $"WHERE Cust_organization LIKE
'%{textBox_Search.Text}%';";
                }
                else
                {
                    queryString = $"SELECT Cust_id, " +
                                  $"Cust_organization AS 'Назва', " +
                                  $"Cust_code AS 'Код ЄДРПОУ', " +
                                  $"Cust_AccauntNumber AS 'Розрах.
рахунок', " +
                                  $"Cust_Orders AS 'Кількість замовл.', "
+
                                  $"Cust_name AS 'ПІБ конт. особи', " +
                                  $"Cust_email AS 'Email', " +
                                  $"Cust_tel AS 'Телефон' " +
                                  $"FROM Customer " +
                                  $"WHERE Cust_code LIKE
'%{textBox_Search.Text}%';";
                }

                SqlDataAdapter adapter = new SqlDataAdapter();
                DataTable table = new DataTable();
                SqlCommand command = new SqlCommand(queryString,
db.getConnection());

                if (db.openConnection() > 0)
                {
```

```
                    adapter.SelectCommand = command;
                    adapter.Fill(table);
                    db.closeConnection();

                    if (table.Rows.Count > 0)
                    {
                        dataGridView_cust.DataSource = table;
                        dataGridView_cust.Columns[0].Visible = false;
                    }
                    else
                    {
                        MessageBox.Show("Замовників не знайдено!
\nЗмініть умови пошуку!", "Пошук",
                                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        return;
                    }

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

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В процесі реалізації додатку «Програмна система обліку проектної організації», а також по її завершенню, виконувалося тестування додатку.

Перелік функцій системи, які тестувалися:

– реєстрація та автентифікація користувача – реєстрація користувача у системі (внесення його логіну та паролю до бази даних);

– валідація користувача, авторизація у систему за допомогою логіну та паролю при чому вхід має відбуватися для трьох різних ролей користувача;

– додавання, редагування та видалення Замовників;

– додавання, редагування та видалення співробітників та відділів;

– додавання, редагування та видалення контрактів, проектів та видів робіт;

– автоматичне формування різноманітних звітів.

На програмній системі були проведені наступні види тестування: інтеграційне тестування даних і бази даних, функціональне тестування, тестування інтерфейсу користувача [12].

Було проведено системне тестування, засноване на коректному сценарії роботи користувача з додатком, що описують повний цикл роботи. В ході системного тестування виявлялися недоліки у інтерфейсі, а також запропоновувалися розширення функціоналу з метою підвищення зручності роботи з додатком. В ході проведення функціонального тестування, виявлялися помилки, засновані на взаємодії функціоналів шляхом реалізації нетривіальних сценаріїв.

4.1 Інтеграційне тестування

Інтеграційне тестування проводилося для тестування методів із запитами до БД, для чого потрібно підключатись до СУБД.

Використовувалися такі технічні прийоми:

– перевірка кожного методу доступу до БД за участю правильних та неправильних даних;

– перевірка БД, щоб переконатися, що дані були заповнені належним чином, чи всі події БД відбулися належним чином;

– перегляд повернених даних, щоб переконатися, що дані були отримані коректно.

Цілісність даних БД була перевірена шляхом виконання наступних дій:

– якщо порожнє або нульове значення можна зберегти або отримати з бази даних;

– якщо введені значення успішно зберігаються у БД.

– якщо дані, що містяться в таблиці БД, можуть бути змінені чи ні;

Перевірка сумісності даних із старим застарілим обладнанням та програмним забезпеченням.

Критеріями завершення вважалося, коли новий користувач коректно реєструється, є можливість авторизуватися на основі вже існуючих даних з переходом до необхідної гілки доступу, проекти та інші дані коректно додаються, а також є можливість їх редагувати, переглядати та видаляти з БД.

4.2 Функціональне тестування

Функціональне тестування проводилося для перевірки відповідності функціональності програми вимогам.

Використовувалися такі технічні прийоми:

– виконання кожного сценарію використання або функції з використанням реальних та неправильних даних (валідних та невалідних) для перевірки наступного:

1) очікувані результати відображаються при використанні дійсних даних;

2) відповідні повідомлення про помилки або попередження виводяться, коли використовуються невірні дані.

Критеріями завершення вважалося, коли основний функціонал, включаючи реєстрацію, авторизацію, додавання, нотифікації, пошук за певними критеріями пройшов успішну перевірку тестами.

4.3 Тестування інтерфейсу користувача

Було проведення тестування графічного інтерфейсу користувача – процес тестування продукту інтерфейсу користувача, для забезпечення його відповідності до даної специфікації. Виконувалося для виявлення помилок у функціональності через інтерфейс, необроблені виключення при взаємодії з інтерфейсом, помилки в інтерфейсі (невідповідність проектної документації, відсутність елементів інтерфейсу, чи навпаки присутність на формах, де вони не передбачені).

Критеріями завершення вважалося, коли всі елементи відображаються на екрані, усі кнопки працюють. Якщо функціонал недоступний для певної ролі користувача – відповідні кнопки, текстові бокси, таблиці та ін. не відображаються.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було виконано аналіз предметної області, виконане моделювання предметної області, розглянуто логіку формування вимог до програмної системи, реалізовано базу даних проходячи етапи від концептуального моделювання предметної області до створення фізичної моделі за допомогою СУБД Microsoft SQL Server та утіліти SQL Server Management Studio.

У результаті роботи було розроблено додаток, що моделює роботу проектної організації. Було реалізовано логіку зберігання і роботи з інформацією, важливість ретельного ставлення до коректного зберігання та обробки даних. Було проведено детальний аналіз, планування, описано процес тестування та його реалізація для програмної системи.

Результатом виконання кваліфікаційної роботи є розробка Windows Forms додатку на тему «Програмна система обліку робіт проектної організації» з використанням реляційних баз даних та логіки об’єктно-орієнтованого програмування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до кваліфікаційної роботи бакалавра за спеціальністю 121 – Інженерія програмного забезпечення, освітньо-професійна програма «Програмна інженерія» для здобувачів усіх форм навчання / Упоряд.: З.В. Дудар, О. В. Вечур, В.І, Каук Р. В Мельнікова., В. Ю Нечволод, О.В. Олійник, Ю. С. Новіков, І.А. Ревенчук, І. П. Сокорчук, І. Ю. Шубін – Харків: ХНУРЕ, 2023. − 53 с (pdf – 522 КБ).

2. Windows Forms Microsoft документація URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0 (дата звернення 11.05.24)

3. SQL Server technical documentation URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16 (дата звернення 03.11.23)

4. Основи з ADO.NET URL: https://itvdn.com/ua/blog/article/basics-ado

5. SQL Server Management Studio (SSMS) documentation URL: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16 (дата звернення 10.06.24)

6. Jira Платформа для управління проектами та відстеження робіт https://community.atlassian.com/t5/Jira-Service-Management-articles/We-are-updating-the-user-interface-for-Assets-in-Jira-Service/ba-p/2312529 (дата звернення 09.05.24)

7. Asana сервіс для організації робочих завдань та управління проектами https://help.asana.com/hc/en-us/articles/13974287963931-Navigating-Asana (дата звернення 09.05.24)

8. Trello – інструмент для керування завданнями та проектами у формі дошок а https://trello.com/

9. Левус, Є.В. Життєвий цикл програмного забезпечення : навчальний посібник / Є.В. Левус, Т.А. Марусенкова, О.О. Нитребич. – Львів : Видавництво Львівської політехніки, 2017. - 207 С.

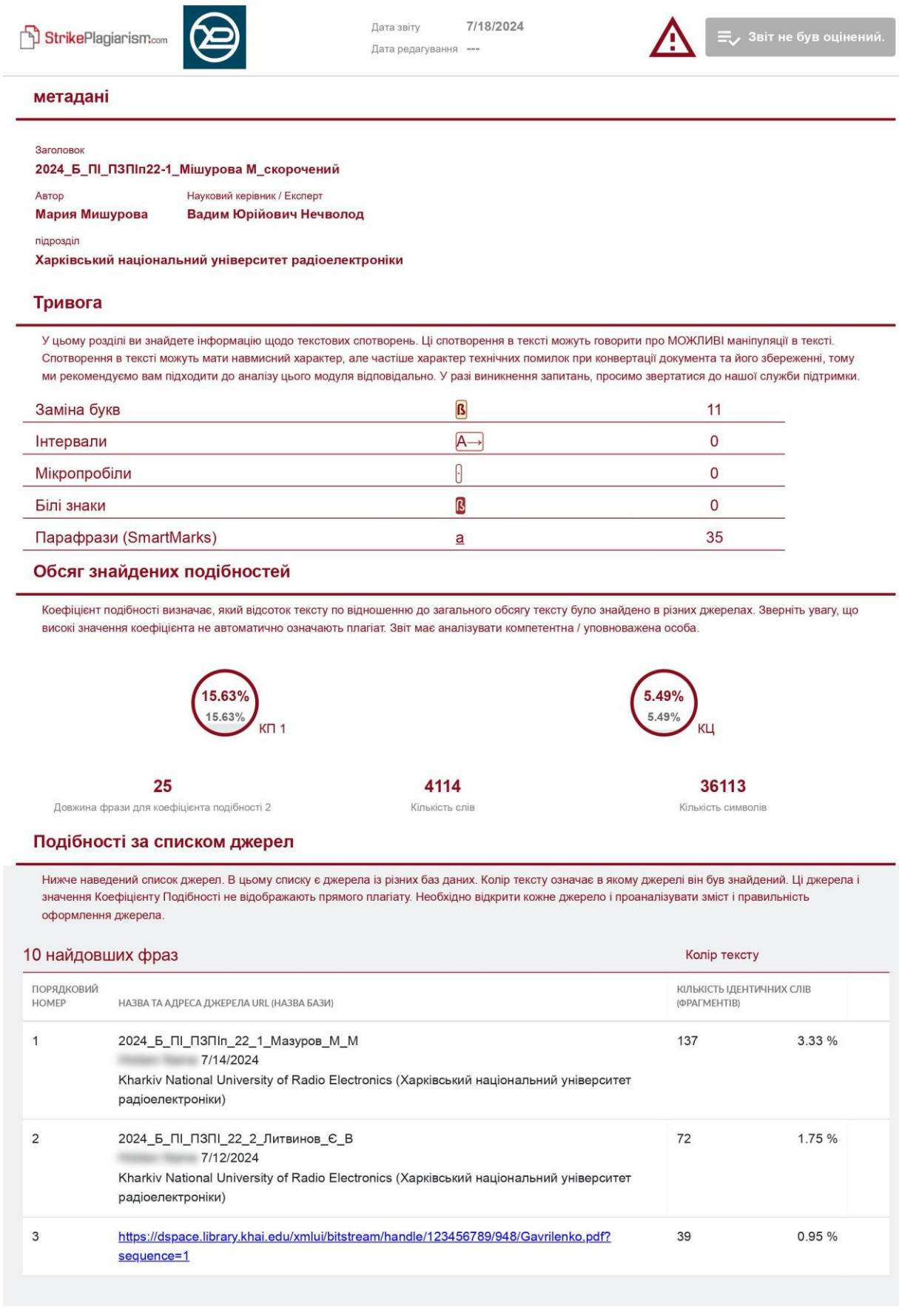
10. Кузьміна, О.О. Архітектура і програмне забезпечення навч. посібник / О.О. Кузьміна. - Дніпропетровськ. : РВВ ДНУ, 2002. - 64 С.

11. Кучеров, Д. П. Інженерія програмного забезпечення : навч. посіб. / Д. П. Кучеров, Є. Б. Артамонов. – Київ : НАУ, 2017. – 386 с.

12. QATestLab. QA термінологія https://training.qatestlab.com/tag/qa-terminology/ (дата звернення 13.07.24).

ДОДАТОК А

Перевірка роботи на унікальність тексту в мережі інтернет та базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/18/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп22-1_Мішурова М_скорочений

Автор
Мария Мишурова

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	ß	11
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	35

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.

15.63% 15.63% КП 1

5.49% 5.49% КЦ

25
Довжина фрази для коефіцієнта подібності 2

4114
Кількість слів

36113
Кількість символів

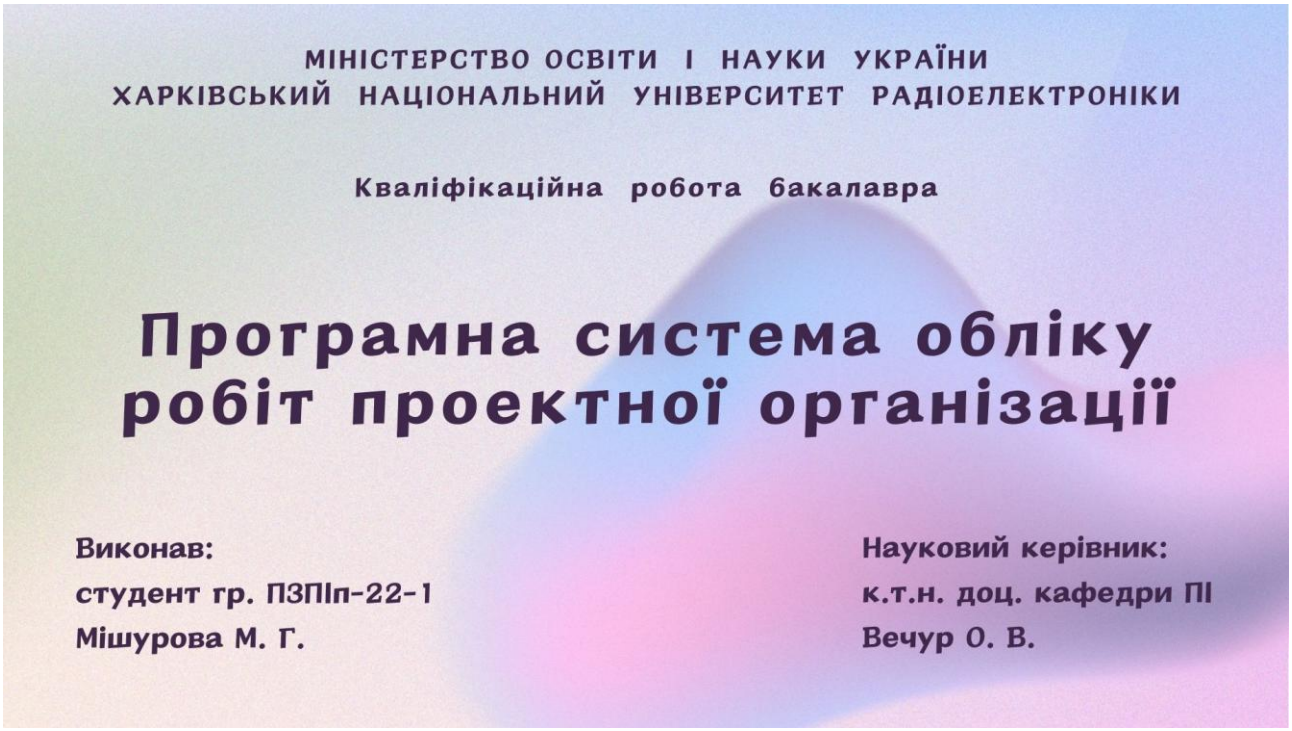
Подібності за списком джерел

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.

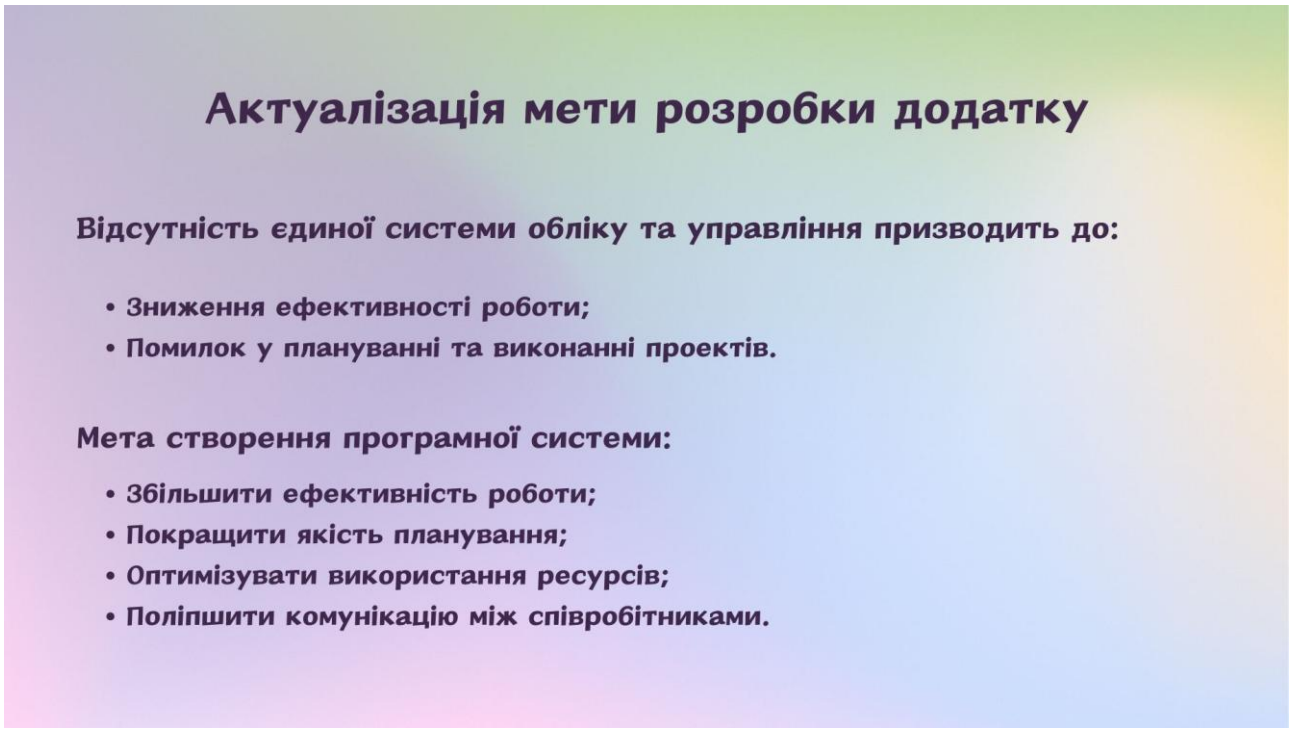
10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	2024_Б_ПІ_ПЗПІп_22_1_Мазуров_М_М [redacted] 7/14/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	137	3.33 %
2	2024_Б_ПІ_ПЗПІ_22_2_Литвинов_Є_В [redacted] 7/12/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	72	1.75 %
3	https://dspace.library.khai.edu/xmlui/bitstream/handle/123456789/948/Gavrilenko.pdf?sequence=1	39	0.95 %

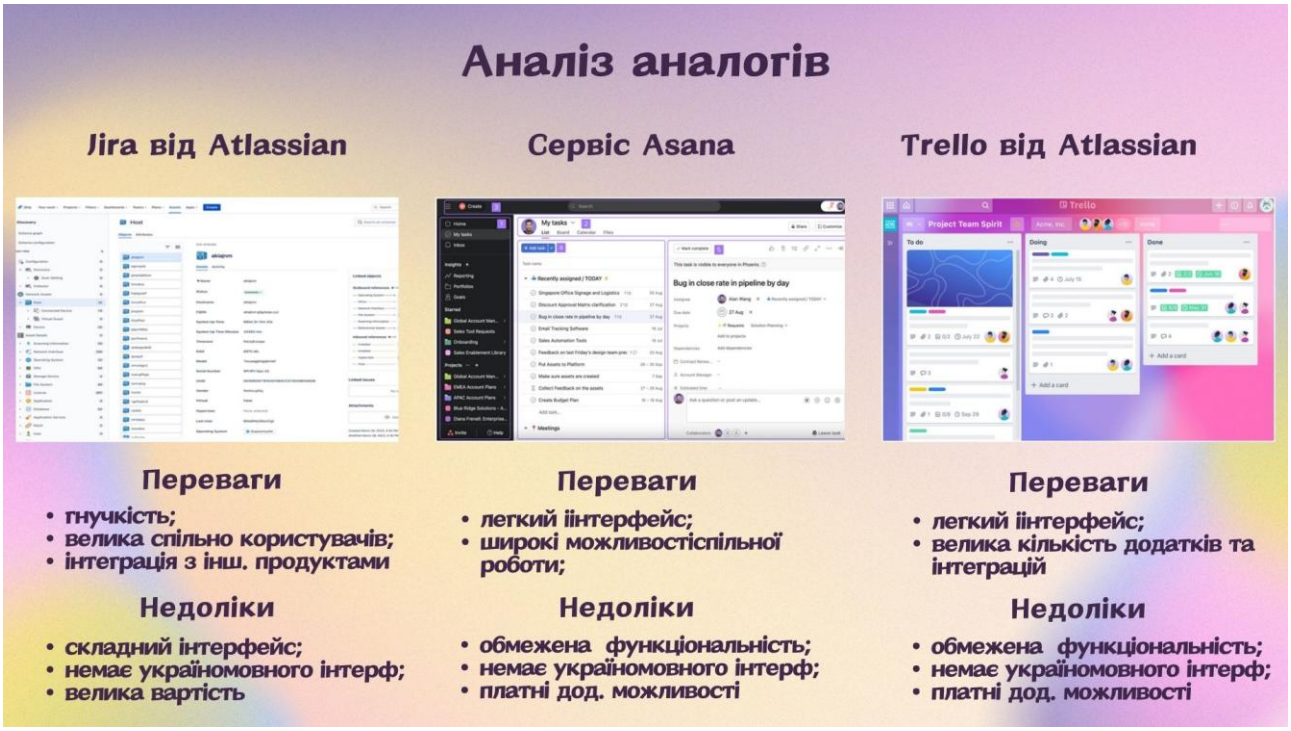
ДОДАТОК Б



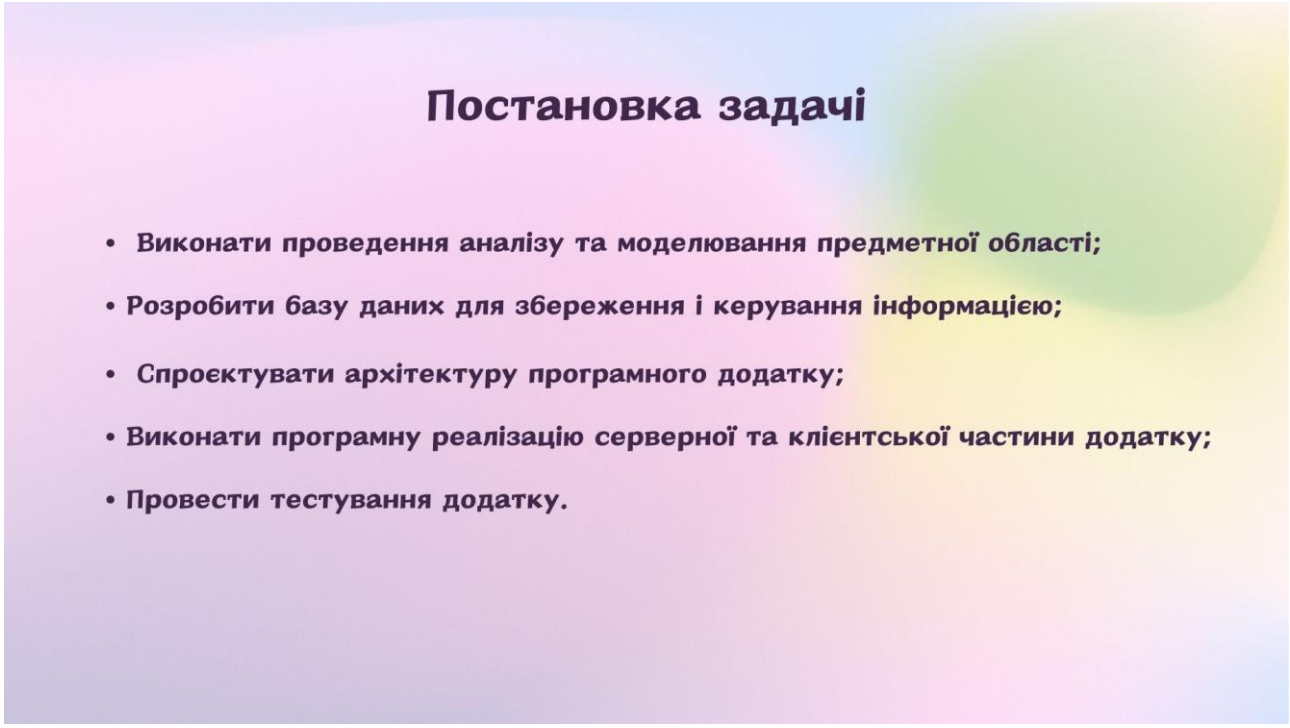
Слайд 1



Слайд 2

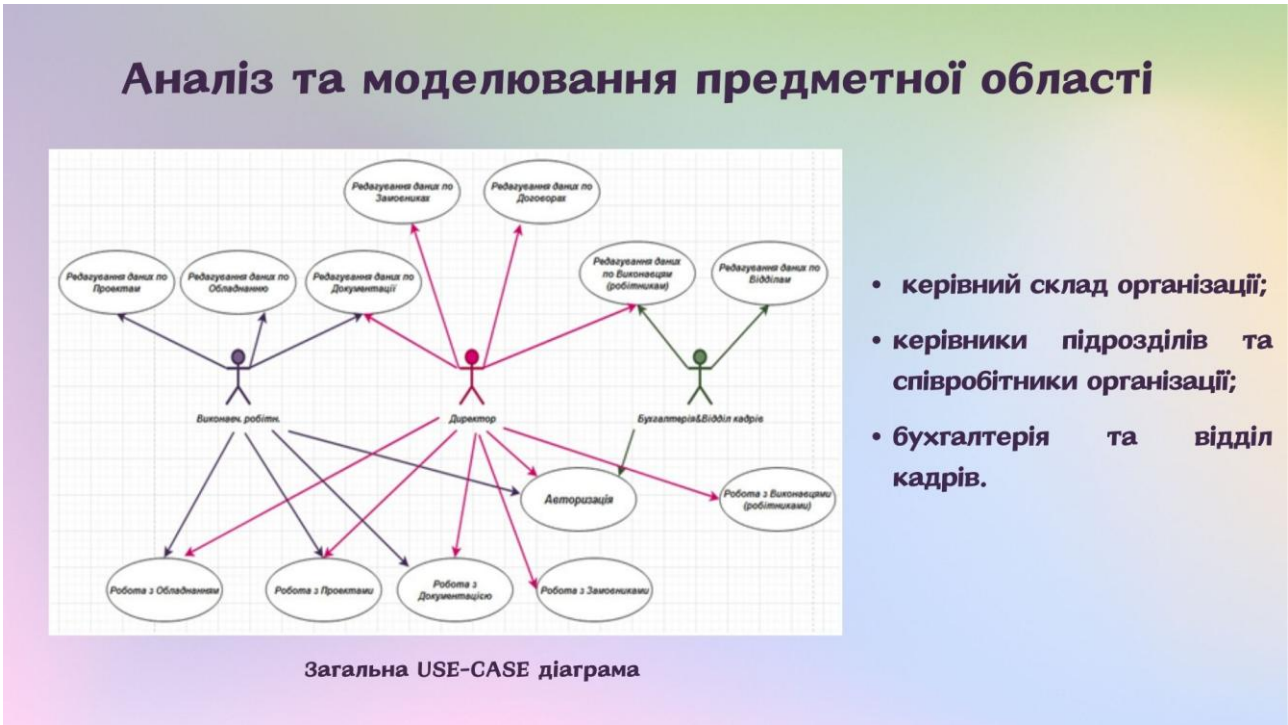


Слайд 3



Слайд 4

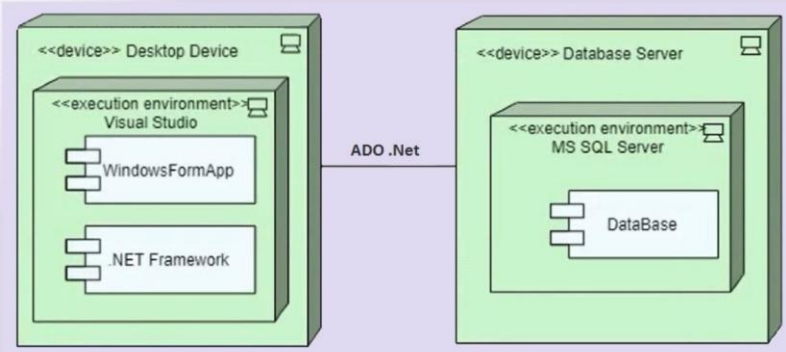
Слайд 5



Слайд 6

Розгортання та запуск системи

- середа розробки MS Visual Studio;
- система управління базами даних MS SQL Server;
- Інтерфейс програмування додатків Windows Forms



Слайд 7

Розробка фізичної моделі БД

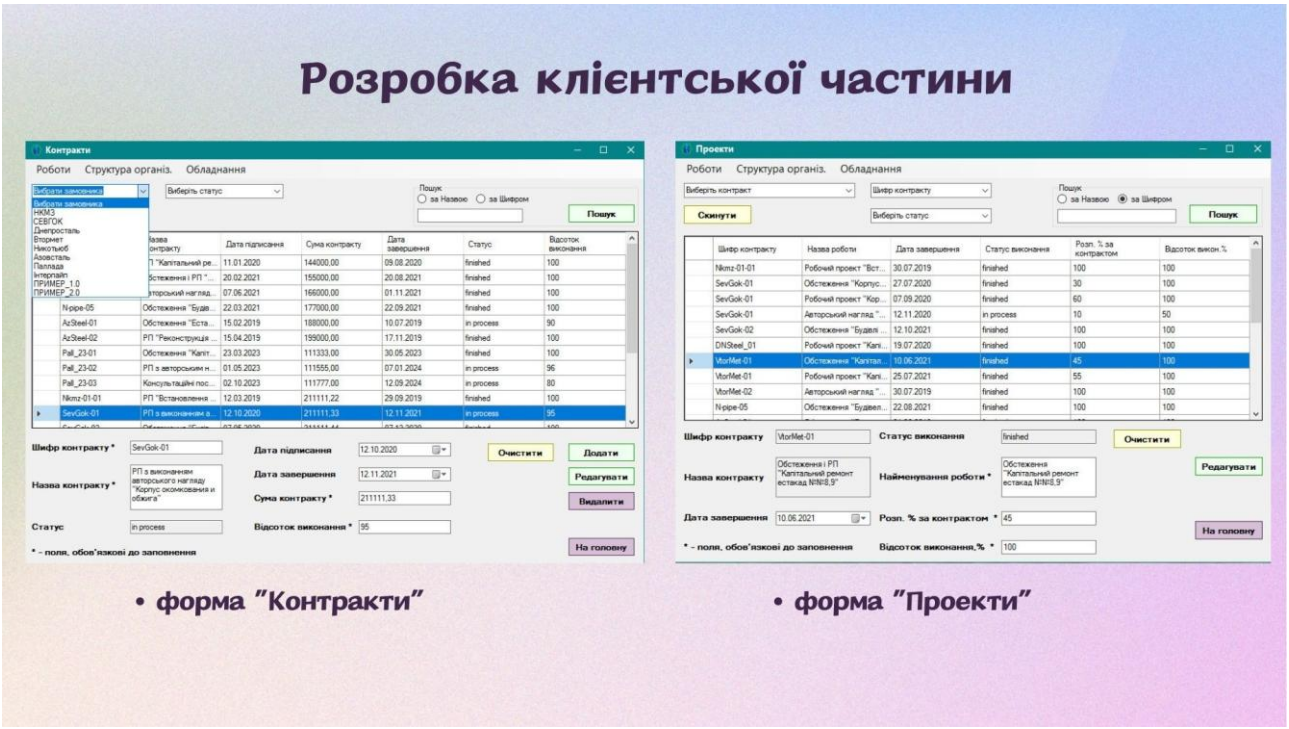
```
CREATE TABLE Customer (
    Cust_id INT IDENTITY(1,1) PRIMARY KEY,
    Cust_organization VARCHAR(255),
    Cust_code VARCHAR(30),
    Cust_AccauntNumber VARCHAR(30),
    Cust_Orders INT,
    Cust_name VARCHAR(50),
    Cust_email VARCHAR(50),
    Cust_tel VARCHAR(20));

CREATE TABLE Contract (
    Contr_id INT IDENTITY(1,1) PRIMARY KEY,
    contr_number VARCHAR(10) NULL,
    contr_name VARCHAR(100) NULL,
    contr_date DATE NULL,
    contr_TotalCost NUMERIC(10, 2) NULL,
    contr_dueDate DATE NULL,
    contr_status VARCHAR(15) NULL,
    cust_id INT,
    CONSTRAINT FK_Customer_Contract
        FOREIGN KEY (cust_id)
        REFERENCES Customer(Cust_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE );
```

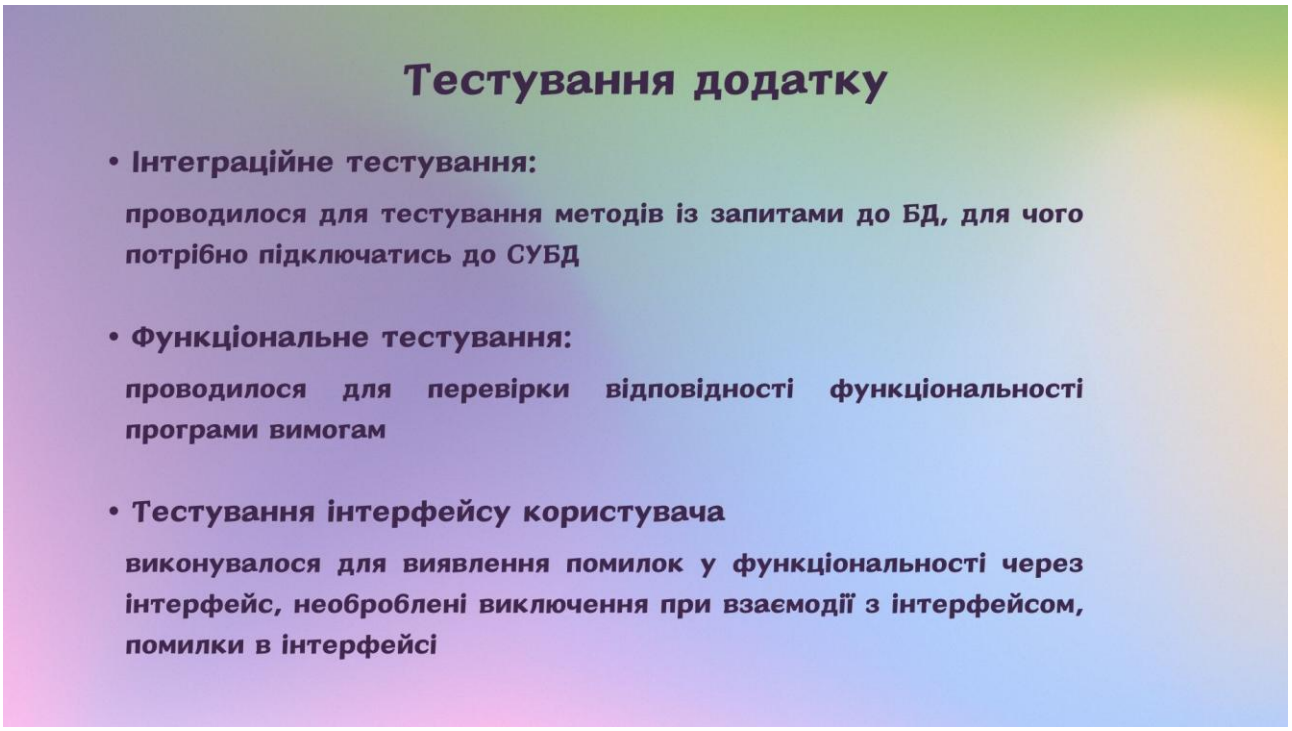
```
CREATE TABLE Department (
    Department_id INT IDENTITY(1,1) NOT NULL
PRIMARY KEY,
    Dep_name VARCHAR(100) NULL,
    Dep_Full_name VARCHAR(100) NULL);

CREATE TABLE Employee (
    Employee_id INT IDENTITY(1,1) NOT NULL
PRIMARY KEY,
    Empl_Full_name VARCHAR(100) NULL,
    Empl_JobTitle VARCHAR(100) NULL,
    Empl_expirience INT NULL,
    Empl_hiringDate DATE NULL,
    Empl_Certificate NCHAR(10) NULL,
    Empl_Salary DECIMAL(12, 2) NULL,
 Login VARCHAR(50) NULL,
 Password VARCHAR(50) NULL,
    Department_id INT NOT NULL,
    FOREIGN KEY (Department_id)
        REFERENCES Department(Department_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

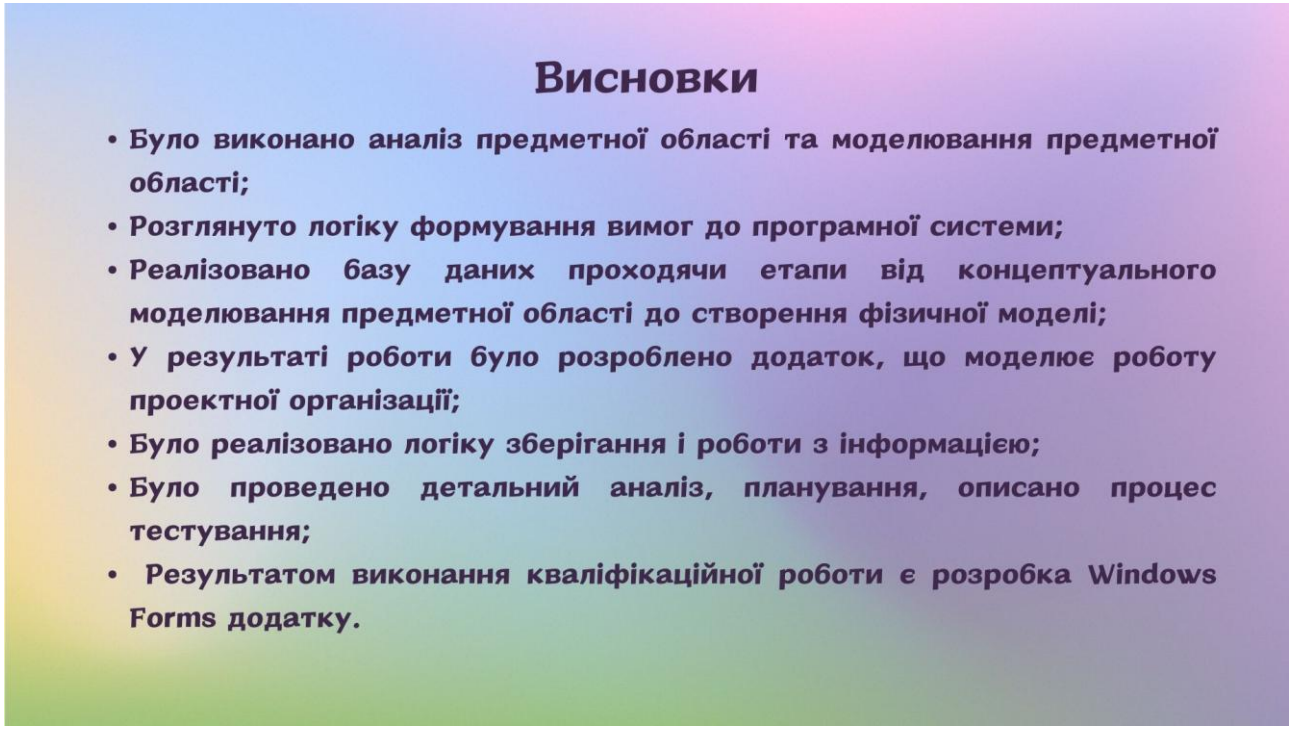
Слайд 8



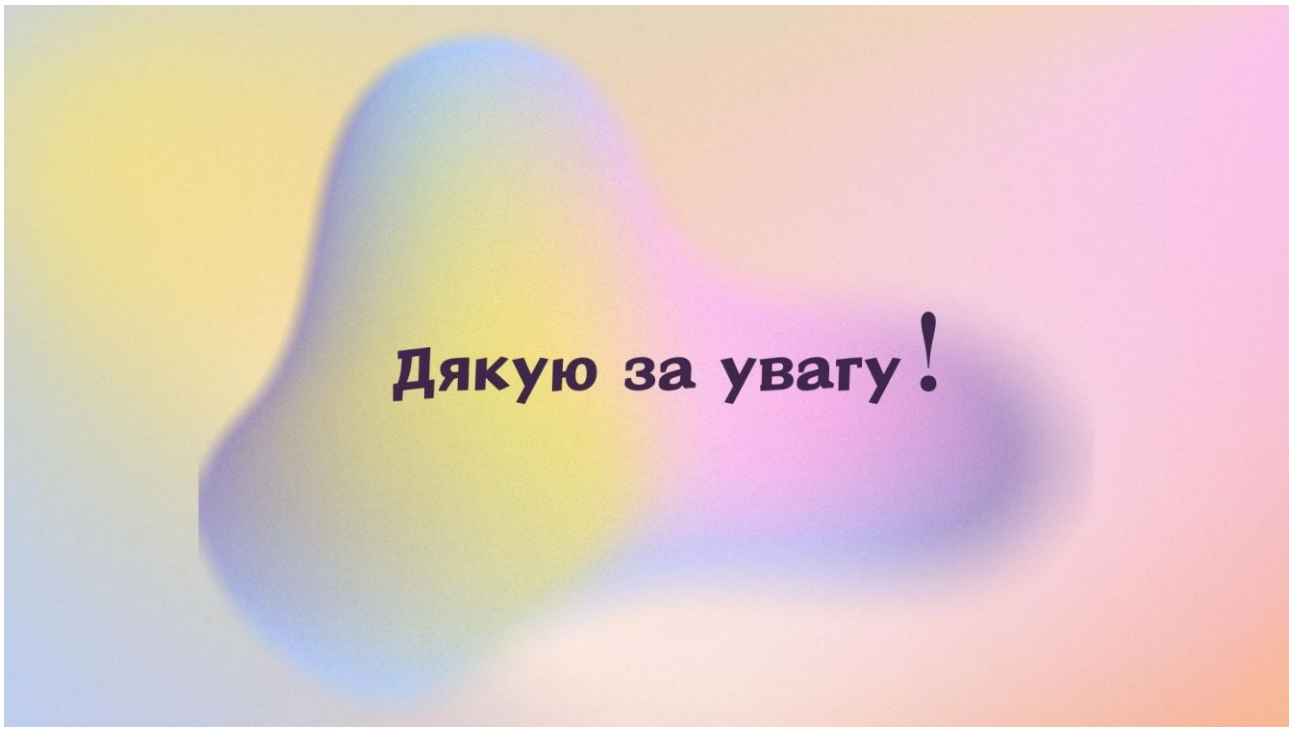
Слайд 9



Слайд 10



Слайд 11



Слайд 12

ДОДАТОК В

Текстовий вивід програмного коду лістинг (на прикладі форми Прилади)

```
public partial class Device : Form
    {
        DataBase db = new DataBase();
        string Device_id = "";
        private string currentUserRole;

        public Device(string role)
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += new EventHandler(Device_Load);
            this.currentUserRole = role;
            CustomizeForRole(role);
        }

        private void CustomizeForRole(string role)
        {
            switch (role)
            {
                case "Директор":
                case "головний спеціаліст":
                case "керівник групи":
                    LoadForDirector();
                    break;
                case "головний бухгалтер":
                case "бухгалтер":
                    LoadForAccounting();
                    break;
                default:
                    LoadForEmployee();
                    break;
            }
        }

        private void dataGridView_dev_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = dataGridView_dev.Rows[e.RowIndex];

                Device_id = row.Cells[0].Value.ToString();
                textBox_FullName.Text = row.Cells[1].Value.ToString();
                textBox_factoryNum.Text = row.Cells[2].Value.ToString();

                if (row.Cells[3].Value != DBNull.Value)
                {
                    DateTime buy_date =
Convert.ToDateTime(row.Cells[3].Value);
                    buyDate.Value = buy_date;
                }
                else
                {
```

```
                    dateTime_certificate.Value = DateTime.Now;
                }

                textBox_price.Text = row.Cells[4].Value.ToString();

                textBox_certificate.Text =
row.Cells[5].Value.ToString();
                textBox_costSertificate.Text =
row.Cells[8].Value.ToString();

                if (row.Cells[6].Value != DBNull.Value)
                {
                    DateTime cert_date =
Convert.ToDateTime(row.Cells[6].Value);
                    dateTime_certificate.Value = cert_date;
                }
                else
                {
                    dateTime_certificate.Value = DateTime.Now;
                }

                if (row.Cells[7].Value != DBNull.Value)
                {
                    DateTime dateTime_cert =
Convert.ToDateTime(row.Cells[7].Value);
                    dateTime_certificate2.Value = dateTime_cert;
                }
                else
                {
                    dateTime_certificate2.Value = DateTime.Now;
                }
            }
        }

        private void Device_Load(object sender, EventArgs e)
        {
            dataGridView_dev.CellClick += new
DataGridViewCellEventHandler(dataGridView_dev_CellClick);

            button_add.Click += new EventHandler(button_add_Click);
            button_edit.Click += new EventHandler(button_edit_Click);
            button_clear.Click += new EventHandler(button_clear_Click);
            button_delete.Click += new
EventHandler(button_Delete_Click);
            button_close.Click += new EventHandler(button_close_Click);

            QueryDevice();

            if (currentUserRole == "Директор" || currentUserRole ==
"головний спеціаліст" || currentUserRole == "керівник групи")
            {
                CheckCertificatesExpiration();
            }
        }

        public void QueryDevice()
        {
            SqlDataAdapter dataAdapter = new SqlDataAdapter();
```

```
            DataTable table = new DataTable();

            string query = @"
     SELECT
        d.Device_id,
        d.Dev_name AS 'Назва приладу',
        d.Dev_factoryNum AS 'Заводський номер',
        d.Dev_buyData AS 'Дата придбання',
        d.Dev_price AS 'Вартість придбання',
        c.Certif_number AS 'Номер сертифіката',
        c.Certif_dataVerificat AS 'Дата верифікації сертифіката',
        c.Certif_varificatDate AS 'Термін дії сертифіката',
        c.Certif_price AS 'Вартість сертифіката'
    FROM
        Device d
    LEFT JOIN
        Certificate c ON d.Device_id = c.Device_id";

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

            try
            {
                if (db.openConnection() > 0)
                {
                    dataAdapter.SelectCommand = command;
                    dataAdapter.Fill(table);
                    dataGridView_dev.DataSource = table;
                    dataGridView_dev.Columns[0].Visible = false;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при виконанні операції:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                db.closeConnection();
            }
        }

        public void LoadForDirector()
        {
            QueryDevice();
        }

        public void LoadForAccounting()
        {
            QueryDevice();
            button_delete.Visible = false;
        }

        public void LoadForEmployee()
        {
            QueryDevice();
            button_add.Visible = false;
            button_delete.Visible = false;
```

```
            button_edit.Visible = false;
        }

        private void button_clear_Click(object sender, EventArgs e)
        {
            ClearTextFields();
        }

        private void ClearTextFields()
        {
            textBox_FullName.Text = string.Empty;
            textBox_factoryNum.Text = string.Empty;
            textBox_price.Text = string.Empty;
            buyDate.Value = DateTime.Now;
            textBox_certificate.Text = string.Empty;
            textBox_costSertificate.Text = string.Empty;
            dateTime_certificate.Value = DateTime.Now;
            dateTime_certificate2.Value = DateTime.Now;
        }

        private void CheckCertificatesExpiration()
        {
            DataTable expiringDevices = GetExpiringDevices();
            if (expiringDevices.Rows.Count > 0)
            {
                string message = "У наступних пристроїв скоро
закінчується термін дії сертифіката:\n\n";
                foreach (DataRow row in expiringDevices.Rows)
                {
                    message += $"{row["Dev_name"]} -
{Convert.ToDateTime(row["Certif_varificatDate"]).ToString("dd.MM.yyyy")}
\n";
                }
                MessageBox.Show(message, "Предупреждение",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private DataTable GetExpiringDevices()
        {
            DataTable table = new DataTable();
            string query = @"SELECT d.Dev_name, c.Certif_varificatDate
                            FROM Device d
                            JOIN Certificate c ON d.Device_id =
c.Device_id
                            WHERE c.Certif_varificatDate <= DATEADD(day,
15, GETDATE())";

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

            try
            {
                if (db.openConnection() > 0)
                {
                    SqlDataAdapter dataAdapter = new
SqlDataAdapter(command);
                    dataAdapter.Fill(table);
```

```
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при виконанні операції:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                db.closeConnection();
            }
            return table;
        }

        private void button_add_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(textBox_FullName.Text) ||
string.IsNullOrEmpty(textBox_price.Text))
            {
                MessageBox.Show("Заповніть всі обов'язкові поля перед
додаванням.", "Попередження", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            try
            {
                string devName = textBox_FullName.Text;
                string factoryNum = textBox_factoryNum.Text;
                DateTime buyD = buyDate.Value;
                decimal totalCost = Decimal.Parse(textBox_price.Text);

                string certNumber = textBox_certificate.Text;
                DateTime certDate = dateTime_certificate.Value;
                DateTime certExpiryDate = dateTime_certificate2.Value;
                decimal certCost =
string.IsNullOrEmpty(textBox_costSertificate.Text) ? 0 :
Decimal.Parse(textBox_costSertificate.Text);

                string queryStringDevice = "INSERT INTO Device
(Dev_name, Dev_buyData, Dev_factoryNum, Dev_price) " +
                                           "VALUES (@Dev_name,
@Dev_buyDate, @Dev_factoryNum, @Dev_price); " +
                                           "SELECT SCOPE_IDENTITY();";

                string queryStringCertificate = "INSERT INTO Certificate
(Device_id, Certif_number, Certif_dataVerificat, Certif_varificatDate,
Certif_price) " +
                                                "VALUES (@Device_id,
@Certif_number, @Certif_dataVerificat, @Certif_varificatDate,
@Certif_price)";

                using (SqlConnection connection = db.getConnection())
                {
                    connection.Open();
                    SqlTransaction transaction =
connection.BeginTransaction();
```

```
                    try
                    {
                        int newDeviceId;
                        using (SqlCommand commandDevice = new
SqlCommand(queryStringDevice, connection, transaction))
                        {

commandDevice.Parameters.AddWithValue("@Dev_name", devName);

commandDevice.Parameters.AddWithValue("@Dev_buyDate", buyD);

commandDevice.Parameters.AddWithValue("@Dev_factoryNum", factoryNum);

commandDevice.Parameters.AddWithValue("@Dev_price", totalCost);

                            newDeviceId =
Convert.ToInt32(commandDevice.ExecuteScalar());
                        }

                        using (SqlCommand commandCertificate = new
SqlCommand(queryStringCertificate, connection, transaction))
                        {

commandCertificate.Parameters.AddWithValue("@Device_id", newDeviceId);

commandCertificate.Parameters.AddWithValue("@Certif_number",
certNumber);

commandCertificate.Parameters.AddWithValue("@Certif_dataVerificat",
certDate);

commandCertificate.Parameters.AddWithValue("@Certif_varificatDate",
certExpiryDate);

commandCertificate.Parameters.AddWithValue("@Certif_price", certCost);
                            commandCertificate.ExecuteNonQuery();
                        }

                        transaction.Commit();
                        MessageBox.Show("Обладнання успішно додано.",
"Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

                        ClearTextFields();
                        QueryDevice();
                    }
                    catch (Exception ex)
                    {
                        transaction.Rollback();
                        MessageBox.Show($"Помилка при виконанні
операції: {ex.Message}", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при виконанні операції:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }
            finally
            {
                db.closeConnection();
            }
        }

        private void button_edit_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(Device_id))
            {
                MessageBox.Show("Оберіть прилад для редагування,
\nвиділівши рядок у таблиці!", "Редагування",
                                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }

            if (string.IsNullOrEmpty(textBox_FullName.Text) ||
string.IsNullOrEmpty(textBox_price.Text))
            {
                MessageBox.Show("Назва приладу та ціна не можуть бути
пустими!", "Редагувати",
                                    MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }

            DialogResult result = MessageBox.Show("Ви точно хочете
виконати редагування?", "Підтвердження редагування",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                string devName = textBox_FullName.Text;
                string factoryNum = textBox_factoryNum.Text;
                DateTime buyD = buyDate.Value;
                decimal totalCost = Decimal.Parse(textBox_price.Text);

                string certNumber = textBox_certificate.Text;
                DateTime certDate = dateTime_certificate.Value;
                DateTime certExpiryDate = dateTime_certificate2.Value;

                decimal certCost =
string.IsNullOrEmpty(textBox_costSertificate.Text) ? 0 :
Decimal.Parse(textBox_costSertificate.Text);

                string queryStringDevice = "UPDATE Device SET Dev_name =
@Dev_name, Dev_buyData = @Dev_buyDate, " +
                                            "Dev_factoryNum =
@Dev_factoryNum, Dev_price = @Dev_price " +
                                            "WHERE Device_id =
@Device_id";

                string queryStringCertificate = "UPDATE Certificate SET
Certif_number = @Certif_number, Certif_dataVerificat =
@Certif_dataVerificat, " +
```

```
                                                    "Certif_varificatDate =
@Certif_varificatDate, Certif_price = @Certif_price " +
                                                    "WHERE Device_id =
@Device_id";

                using (SqlConnection connection = db.getConnection())
                {
                    try
                    {
                        connection.Open();
                        SqlTransaction transaction =
connection.BeginTransaction();

                        SqlCommand commandDevice = new
SqlCommand(queryStringDevice, connection, transaction);
                        SqlCommand commandCertificate = new
SqlCommand(queryStringCertificate, connection, transaction);

                        commandDevice.Parameters.Clear();

commandDevice.Parameters.AddWithValue("@Device_id", Device_id);

commandDevice.Parameters.AddWithValue("@Dev_name", devName);

commandDevice.Parameters.AddWithValue("@Dev_buyDate", buyD);

commandDevice.Parameters.AddWithValue("@Dev_factoryNum", factoryNum);

commandDevice.Parameters.AddWithValue("@Dev_price", totalCost);

                        commandCertificate.Parameters.Clear();

commandCertificate.Parameters.AddWithValue("@Device_id", Device_id);

commandCertificate.Parameters.AddWithValue("@Certif_number",
certNumber);

commandCertificate.Parameters.AddWithValue("@Certif_dataVerificat",
certDate);

commandCertificate.Parameters.AddWithValue("@Certif_varificatDate",
certExpiryDate);

commandCertificate.Parameters.AddWithValue("@Certif_price", certCost);

                        try
                        {
                            commandDevice.ExecuteNonQuery();
                            commandCertificate.ExecuteNonQuery();
                            transaction.Commit();

                            ClearTextFields();
                            QueryDevice();
                        }
                        catch (Exception ex)
                        {
                            transaction.Rollback();
```

```
                            MessageBox.Show($"Помилка при виконанні
операції: {ex.Message}", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        }
                        db.closeConnection();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show($"Помилка при відкритті
з'єднання: {ex.Message}", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                    finally
                    {
                        db.closeConnection();
                    }
                }
            }
        }

        private void button_Delete_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(Device_id))
            {
                MessageBox.Show("Оберіть прилад для видалення,
\nвиділивши рядок у таблиці!", "Видалення",
                                    MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }

            if (MessageBox.Show("Ви точно хочете видалити дані?",
"Підтвердження видалення",
                 MessageBoxButtons.YesNo, MessageBoxIcon.Warning) ==
DialogResult.No)
            {
                return;
            }

            string queryStringCertificate = "DELETE FROM Certificate
WHERE Device_id = @Device_id;";
            string queryStringDevice = "DELETE FROM Device WHERE
Device_id = @Device_id;";

            using (SqlConnection connection = db.getConnection())
            {
                try
                {
                    connection.Open();
                    SqlTransaction transaction =
connection.BeginTransaction();

                    try
                    {
                        using (SqlCommand commandCertificate = new
SqlCommand(queryStringCertificate, connection, transaction))
                        {
```

```
commandCertificate.Parameters.AddWithValue("@Device_id", Device_id);
                            commandCertificate.ExecuteNonQuery();
                        }

                        using (SqlCommand commandDevice = new
SqlCommand(queryStringDevice, connection, transaction))
                        {

commandDevice.Parameters.AddWithValue("@Device_id", Device_id);
                            commandDevice.ExecuteNonQuery();
                        }

                        transaction.Commit();

                        ClearTextFields();
                        QueryDevice();
                    }
                    catch (Exception ex)
                    {
                        transaction.Rollback();
                        MessageBox.Show($"Помилка при виконанні
операції: {ex.Message}", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка при відкритті з'єднання:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    db.closeConnection();
                }
            }
        }
```