ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання мовою C#»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ольга ПЕРМЯКОВА
(підпис)

Виконала: здобувачка вищої освіти групи ПД-43

Ольга ПЕРМЯКОВА

Керівник: Ігор ГАМАНЮК
старший викладач кафедри ІПЗ

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пермяковій Ользі Олександрівні

1. Тема кваліфікаційної роботи: «Розробка застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання мовою C#»
керівник кваліфікаційної роботи старший викладач кафедри ІПЗ Ігор ГАМАНЮК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про телекомунікаційне обладнання, міждержавні стандарти технічного обслуговування.

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

1. Аналіз інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання, визначення потреб і вимог
2. Засоби розробки програмного продукту
3. Моделювання та проектування застосунку
4. Програмна реалізація застосунку

5. Тестування застосунку

5. Перелік графічного матеріалу: *презентація*

1. Аналіз інструментальних засобів.
2. Вимоги до застосунку.
3. Програмні засоби реалізації.
4. Діаграма прецедентів.
5. Діаграма пакетів.
6. Діаграма послідовності.
7. Діаграма класів працівників.
8. Діаграма класів виконаних обслуговувань.
9. ER-діаграма бази даних.
10. Екранні форми.
11. Відео роботи застосунку
12. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд засобів розробки програмного продукту	14.03-20.03.2024	
4	Моделювання та проектування застосунку	21.03-02.04.2024	
5	Програмна реалізація застосунку	03.04-19.04.2024	
6	Тестування застосунку	20.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач(ка) вищої освіти _____ Ольга ПЕРМЯКОВА
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 40 стор., 2 табл., 35 рис., 13 джерел.

Мета роботи – спрощення процесу обліку періодичності технічного обслуговування телекомунікаційного обладнання шляхом впровадження застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Об'єкт дослідження – процес обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Предмет дослідження – застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Короткий зміст роботи: В роботі проаналізовано предметну область обліку технічного обслуговування телекомунікаційного обладнання. Проаналізовано інструментальні засоби для обліку технічного обслуговування телекомунікаційного обладнання: Microsoft Excel, Windows Notepad, Hippo CMMS. Розроблено застосунок з обліку періодичності технічного обслуговування телекомунікаційного обладнання з використанням мови C#, середовища розробки Visual Studio та платформи .NET Framework. Для створення інтерфейсу користувача було використано графічну підсистему WinForms. В якості системи керування базою даних було обрано SQLite та EntityFramework Core для взаємодії з нею. Програмно реалізовані ключові функціональні можливості, зокрема: редагування, видалення та додавання нового обладнання, працівників, виконаних технічних обслуговувань та видів технічних обслуговувань, а також пошук та фільтрація даних для відображення. Проведено функціональне тестування додатку.

Сферою використання застосунку є облік технічного обслуговування телекомунікаційного обладнання середніми та малими компаніями і фірмами, індивідуальними підприємцями, які мають у своєму розпорядженні

телекомунікаційне обладнання, що підлягає технічному обслуговуванню.

КЛЮЧОВІ СЛОВА: ТЕЛЕКОМУНІКАЦІЙНЕ ОБЛАДНАННЯ, ТЕХНІЧНЕ ОБСЛУГОВУВАННЯ, VISUAL STUDIO, C#, .NET FRAMEWORK, WINFORMS, ENTITYFRAMEWORK CORE, SQLITE, ЗАСТОСУНОК.

ЗМІСТ

ВСТУП

Людство, протягом свого існування, завжди намагалося спростити вирішення питань в швидкому досягненні бажаних результатів. Це твердження буде вірним в будь-якій сфері діяльності, як окремих індивідуумів, так і груп людей. Технологічний прогрес надав можливість використовувати свої досягнення не тільки великим корпораціям, а і середнім, малим фірмам, індивідуальним підприємцям. Більш яскраво розвиток технологій всі спостерігають останні десятиліття в секторі інформаційних технологій. Це стосується, як удосконалень існуючих програмних продуктів, так і появи великого різноманіття програмних продуктів та застосунків для спрощення усіх сфер життєдіяльності людства.

Спостереження за ринком програмних продуктів наводять на думку, що більшість програмних продуктів і застосунків створюються великими фірмами, не тільки для спрощення життєдіяльності людства, а і, банально, для отримання прибутку. Це накладає відбиток великої універсальності на застосунки і програми створені для користувачів. І це є правильно з точки зору намагання великих виробників, щоб їх програмними продуктами і застосунками користувалось якомога більше користувачів фірм. Але варто зазначити, що більшість функціоналу і можливостей великих програмних продуктів не використовується повним обсягом користувачами. Також, чітко прослідковуються тенденції збільшення потреб хардверних ресурсів при удосконаленні та розвитку програмного забезпечення. В цій ситуації фірми маленького та середнього бізнесу, у яких немає великих оборотних капіталів для розвитку, часто не мають змоги рухатись у ногу з часом та постійно оновлювати і купувати нові програмні застосунки, удосконалювати своє серверне обладнання.

Їм на допомогу має прийти сучасне програмне забезпечення яке може вирішити для них основні дві задачі: по перше, невеликі вимоги до серверного устаткування, по друге, рішення з потрібних споживачу завдань без громіздких

інструментів і надбудов. Простіше кажучи, у застосунку чи програмі повинно бути тільки те, що вирішує завдання споживача і нічого зайвого.

Актуальність теми: Як можна зрозуміти, з зазначеного вище, практично всі суб’єкти господарювання використовують для спрощення та вирішення своїх завдань різноманітну комп’ютерну техніку та мережеве обладнання. Постійне використання мережевого обладнання і комп’ютерної техніки, для їх безперебійної роботи, забезпечується завдяки вчасному та професійному технічному обслуговуванні. Для своєчасного та професійного обслуговування фахівцям, які їм займаються, потрібен невеликий, але зручний інструмент який дає змогу спростити ведення обліку технічного обслуговування обладнання.

Об’єкт дослідження – процес обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Предмет дослідження – застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Мета роботи – спрощення процесу обліку періодичності технічного обслуговування телекомунікаційного обладнання шляхом впровадження застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Завданням роботи є проектування та розробка застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Методика дослідження: Перш за все, було проведено аналіз та порівняння існуючих інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання. На основі цього аналізу були визначені основні вимоги, які мають бути реалізовані. Далі, визначено архітектуру та засоби розробки, що використовуватимуться у розробленому застосунку.

Практична значущість результатів полягає у можливості використання розробленого застосунку середніми та малими компаніями і фірмами, індивідуальними підприємцями, які мають у своєму розпорядженні телекомунікаційне обладнання, що підлягає технічному обслуговуванню.

1 АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ОБЛІКУ ПЕРІОДИЧНОСТІ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ ТЕЛЕКОМУНІКАЦІЙНОГО ОБЛАДНАННЯ, ВИЗНАЧЕННЯ ПОТРЕБ І ВИМОГ

1.1 Microsoft Excel

Microsoft Excel[5] – це універсальна програма для роботи з електронними таблицями та є одним з найвідоміших засобів для проведення аналізу та обробки даних. Microsoft Excel, дозволяє легко опрацьовувати та візуалізувати необхідні дані за допомогою діаграм і графіків, що полегшує їх розуміння і обробку. Програма має безліч вбудованих функцій та інструментів, робота з якими вимагає від користувача певних знань та навичок використання.

Переваги:

– інтерфейс excel знайомий більшості людей, що робить його легким для освоєння та використання;

– ви можете створювати прості таблиці для відстеження інформації про обладнання та виконане технічне обслуговування без необхідності знання складних формул або макросів;

– існує багато навчальних посібників та ресурсів онлайн, які допоможуть вам розпочати роботу з excel;

– excel пропонує широкий спектр функцій, які можна використовувати для налаштування бази даних відповідно до ваших потреб;

– ви можете створювати власні поля, формули та звіти для аналізу даних;

– ви можете використовувати функції форматування, щоб зробити ваші таблиці більш візуально привабливими та зручними для сприйняття;

– ви можете імпортувати та експортувати дані з інших джерел, таких як бази даних або електронні таблиці;

– excel доступний на більшості комп'ютерів та мобільних пристоїв;

– існує безкоштовна версія excel, яка пропонує багато основних функцій.

Недоліки:

– excel не призначений для керування великими та складними базами даних;

– зі збільшенням обсягу даних excel може стати повільним і незручним для використання;

– таблиці excel можуть стати громіздкими та складними для навігації, коли в них багато даних;

– використання excel вимагає певних знань та навичок;

– неспеціалізований інтерфейс.

1.2 Windows Notepad

Windows Notepad[9] – це простий, електронний блокнот для зберігання різноманітної інформації, від своїх власних думок до важливих даних. З 1985 року ця програма вбудована в усі версії Microsoft Windows. Він підтримує такі кодування тексту, як UTF-8, ANSI, Unicode. Це інструмент з мінімальними функціями інтерфейсу користувача, що є зручним для людей з мінімальними навичками. Notepad використовують для легкого та швидкого створення, редагування та збереження текстової інформації, у випадках, коли вона не потребує додаткового форматування та оформленння.

Переваги:

– не потребує жодних спеціальних знань або навичок;

– записи в Notepad можна робити швидко і легко, без необхідності використовувати складні інструменти або меню;

– доступний на будь-якому комп'ютері, без необхідності встановлювати додаткове програмне забезпечення.

Недоліки:

– Notepad не пропонує жодної структури для даних, що може ускладнити їх пошук та аналіз;

– не має жодних вбудованих функцій для аналізу даних або створення звітів;

– редагування та оновлення інформації в блокноті може бути складним і схильним до помилок;

– неспеціалізований інтерфейс.

1.3 Hippo CMMS

Hippo CMMS[4] – це програмне забезпечення, яке було створено для допомоги великим та середнім підприємствам керувати технічним обслуговуванням свого обладнання. Воно може бути корисним інструментом для організованого та ефективного зберігання даних про наявне обладнання та його технічне обслуговування.

Переваги:

– дозволяє створювати та підтримувати інвентаризацію обладнання, включаючи інформацію про місцезнаходження, модель, серійний номер, гарантію тощо;

– дозволяє планування профілактичного та виправного технічного обслуговування обладнання;

– дозволяє створювати та призначати завдання на технічне обслуговування технікам;

– допомагає відстежувати хід робіт з технічного обслуговування та забезпечувати їх своєчасне виконання;

– пропонує різноманітні звіти, які допомагають отримати уявлення про стан обладнання та програму технічного обслуговування;

– має веб-інтерфейс, що дозволяє легко отримати доступ до системи з будь-якого пристрою з підключенням до інтернету.

Недоліки:

– це платне програмне забезпечення;

– для налаштування та використання всіх функцій може знадобитися певне навчання та налаштування;

– потребує підключення до інтернету для доступу до системи;

– не підтримує українську мову, як мову інтерфейсу;

– багато функцій, у яких немає потреби для користувачів з маленьких фірм та підприємств, які, в свою чергу, спричиняють підвищене використання комп’ютерних ресурсів;

– програмне забезпечення загального використання, неспеціалізоване під телекомунікаційне обладнання.

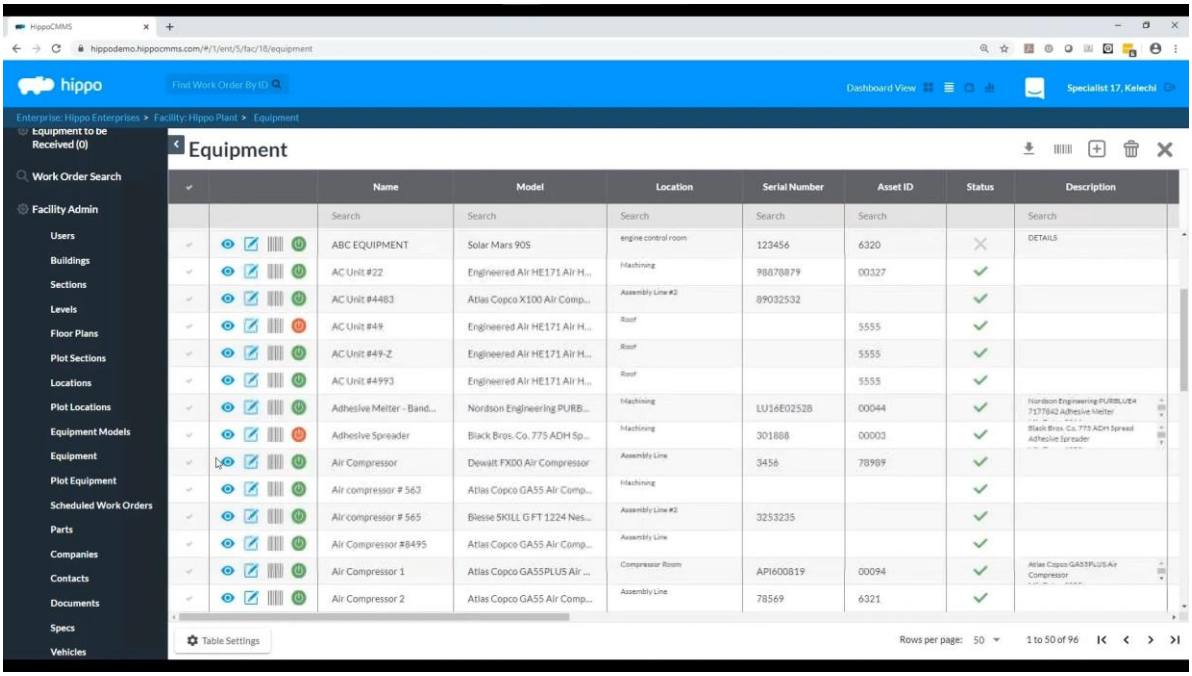


Рис. 1.1 Сторінка перегляду наявного обладнання Hippo CMMS

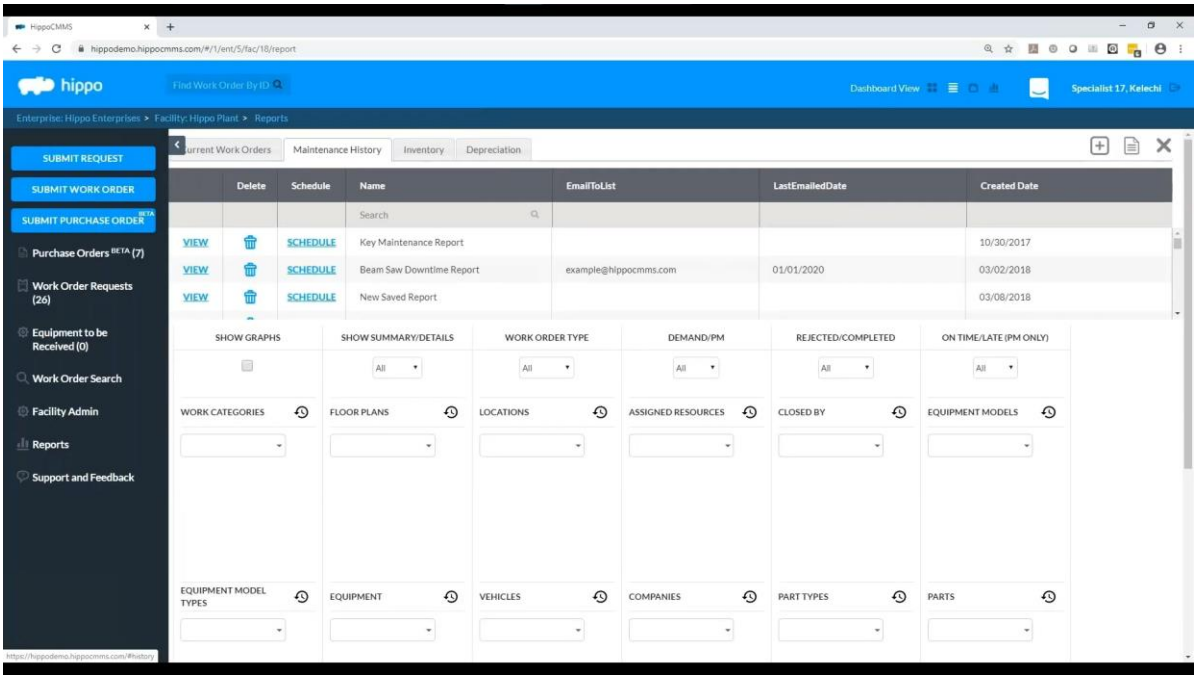


Рис. 1.2 Сторінка перегляду виконаних технічних обслуговувань Hippo CMMS

Таблиця 1.1 представляє зведені результати аналізу інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Таблиця 1.1

Результати аналізу розглянутих інструментальних засобів

	Microsoft Excel	Windows Notepad	Hippo CMMS	Розроблений застосунок
Українська локалізація	+	+	-	+
Можливість доступу без підключення до інтернету	+	+	-	+
Автоматичне структурування даних	-	-	+	+
Фільтрування проведених технічних обслуговувань та телекомунікаційного обладнання, які відображаються користувачу	-	-	-	+

Продовження таблиці 1.1

Результати аналізу розглянутих інструментальних засобів

	Microsoft Excel	Windows Notepad	Hippo CMMS	Розроблений застосунок
Пошук обладнання за інвентарним номером та моделлю	+	-	+	+
Пошук проведеного технічного обслуговування за датою проведення	+	-	+	+

1.4 Визначення потреб і вимог

На основі проведеного аналізу існуючих інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання було визначено наступні функціональні та нефункціональні вимоги:

Функціональні вимоги:

– внесення в систему даних про телекомунікаційне обладнання (тип, модель, інвентарний номер, MAC адреса, тощо);

– додавання нових виконаних технічних обслуговувань;

– можливість перегляду усього доданого телекомунікаційного обладнання;

– можливість перегляду виконаних технічних обслуговувань;

– можливість перегляду повної інформації про обрану одиницю обладнання;

– засоби для роботи з типами технічних обслуговувань та даними користувачів (додавання, видалення, редагування);

– пошук необхідної одиниці телекомунікаційного обладнання за інвентарним номером та моделлю;

– пошук виконаного технічного обслуговування за датою проведення;

– можливість фільтрування за типом телекомунікаційного обладнання та видом технічного обслуговування, при перегляді виконаних технічних обслуговувань.

Нефункціональні вимоги:

– українська локалізація;

– застосунок повинен відповідати на запити користувачів в межах 2 секунд;

– підтримка платформи Windows 10 і вище.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Середовище розробки Microsoft Visual Studio 2022

Visual Studio - це інтегроване середовище розробки (IDE) від Microsoft, яке використовується для створення програмного забезпечення на різних мовах програмування, таких як C#, C++, Python, Java та інші. Воно має широкі можливості для написання, налагодження та тестування коду. У Visual Studio є різноманітні інструменти для підтримки розробки, такі як автодоповнення коду, інтегровані системи керування версіями, візуальні дизайнери інтерфейсу користувача та багато іншого[7].

Особливості Visual Studio:

- підтримка мов програмування C#, C++, Python, Java та багато інших;
- автоматичне доповнення коду, рефакторинг, інтегрована система керування версіями та інші інструменти для полегшення написання коду;
- покрокове виконання коду, установка точок зупинки, перегляд змінних та багато іншого для виявлення та виправлення помилок;
- створення та запуск модульних, інтеграційних та функціональних тестів для перевірки якості вашого коду;
- візуальні дизайнери інтерфейсу користувача, інструменти для роботи з базами даних, можливість публікації веб-застосунків та багато іншого.

Також Visual Studio пропонує кілька технологій та інструментів для створення графічного інтерфейсу користaвача на різних платформах. Для створення десктопних додатків це Windows Form (WimForm), WPF (Windows Presentation Foundation) та UWP (Universal Window Platform), який, в свою чергу, дозволяє розробку інтерфейсу для всіх пристроїв під управлінням Windows 10. Для мобільних додатків IOS та Android це Xamarin.

Рис. 2.1 Логотип Microsoft Visual Studio 2022

2.2 Entity Framework Core

Entity Framework Core - це структура об’єктно-реляційного відображення (ORM) для ADO.NET. Він дозволяє реалізувати зіставлення відносин об’єктів реляційної бази даних з об’єктами мови програмування. Entiti framework Core була спеціально розроблена для платформ .NET Core та .NET 5 і вище. Підтримує різні провайдери баз даних і може працювати з реляційними базами даних, такими як SQL Server, MySQL, PostgreSQL, SQLite, а також з NoSQL[3].

В EF Core доступ до даних реалізується через модель. Під моделлю мається на увазі набір класів сутностей та клас контексту об’єкта, який, в свою чергу, представляє взаємодію з базою даних. EF Core підтримує такі підходи, як Database First та Code First. У першому випадку спочатку створюється база даних та на її основі модель і клас контексту, у другому – спочатку вручну прописується модель даних та контекст, а потім, на основі цієї моделі, створюється база даних.

Рис. 2.2 Логотип Entity Framework Core

2.3 .NET framework

.NET Framework - це платформа для розробки програмного забезпечення, яку створила компанія Microsoft. Вона забезпечує середовище для запуску та розгортання різноманітних додатків, таких як веб-додатки, десктопні застосунки, мобільні додатки та Інтернет-сервіси[6]. Основними компонентами .NET Framework є:

– CLR (Common Language Runtime) − це віртуальна машина, що виконує код .NET. Вона управляє пам'яттю, виконанням коду, обробкою винятків та іншими аспектами виконання програм на платформі .NET;

– базова бібліотека класів .NET (BCL) − Це набір класів, який надає реалізацію ряду функцій: мережевий доступ, файловий доступ, роботу з базами даних тощо. BCL дає доступ до широкого спектру можливостей для розробників;

– мови програмування, такі як C#, VB.NET, F# та деякі інші;

– ASP.NET для створення веб-додатків та сервісів;

– ADO.NET для роботи і доступу до даних.

Рис. 2.3 Логотип .NET Framework

2.4 Windows Forms

Технологія базована на мові програмування C# або Visual Basic .NET, користується бібліотекою класів .NET Framework або .NET Core. Windows Forms створена для розробки програмного забезпечення для платформи Microsoft

Windows, яка дозволяє створювати графічні користувацькі інтерфейси (GUI) для програм. Технологія дозволяє швидко створювати десктопні додатки для Windows, включаючи програми зі складними інтерфейсами, ігри, утиліти, програми для роботи з базами даних та багато іншого[8].

Переваги WinForms:

– WinForms має чітку структуру та зрозумілий синтаксис, що робить його доступним навіть для початківців;

– WinForms надає широкий набір елементів управління, таких як кнопки, текстові поля, списки, таблиці та багато іншого, що дозволяє створювати різноманітні інтерфейси;

– WinForms можна розширювати та налаштовувати за допомогою власних елементів управління та кодів, що робить його гнучким інструментом для створення унікальних рішень;

– WinForms сумісний з усіма версіями Windows.

Рис. 2.4 Логотип WinForms

2.5 Об’єктно-орієнтована мова програмування C#

C# мова програмування з безпечною формою типізації для платформи .NET. Мова є об’єктно орієнтованою, що означає що використовуються наступні механізми:

1. Успадкування. Створення нового класу об’єктів шляхом додавання нових елементів до вже наявного класу.

2. Інкапсуляція. Суттєво спрощує модифікацію програмного забезпечення. По своїй суті це приховування деталей реалізації, яке дозволяє вносити зміни в частини програми безболісно для інших її частин.

3. Поліморфізм. Властивість деяких методів батьківського класу замінюватись новими, що реалізують специфічні для даного нащадка дії.

Переваги C#:

– мова дозволяє створювати різноманітні програми починаючи від ігор і закінчуючи веб-сайтами;

– мова перейняла дуже багато від інших мов (C++, Object Pascal та ін.), тому досвід роботи з іншими мовами програмування дозволить легше вивчити C#;

– .NET Framework містить багато інструментів для полегшення розробки.

Сфери застосування: розробка віконних додатків (Windows Forms, WPF), створення веб-сервісів і сайтів (ASP.NET), розробка мобільних додатків (Xamarin), ігри (Unity).

Рис. 2.5 Логотип C#

2.6 SQLite

SQLite – це полегшена реляційна система керування базами даних. Вона є локальною(вбудованою) базою даних, що не потребує окремого серверу.

Серед основних переваг використання SQLite можна виділити такі:

– так як база даних не потребує окремого серверу вона зберігається в одному крос-платформному файлі на тому комп’ютері, на якому використовується застосунок та постачається одразу разом із програмою;

– крос-платформність: SQLite підтримує майже усі існуючі типи операційних систем(наприклад, Windows, MacOS, Linux, Android, IOS);

– SQLite створена за стандартом SQL-92, що спрощує роботу з даними та забезпечує її сумісність з більшістю інших баз даних;

– широкий спектр інтеграцій з різними мовами програмування і фреймворками;

– доступ до бази даних SQLite не залежить від наявності підключення до мережі.

Варто зазначити, що SQLite є менш вразливою до атак ніж клієнт-серверні бази даних. Оскільки SQLite працює локально та використовує простий файловий формат , її безпека не залежить від мережевих атак[13].

Рис. 2.6 Логотип SQLite

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ЗАСТОСУНКУ

3.1 Модель предметної галузі

Модель предметної галузі − це абстрактне представлення певної предметної галузі, що описує її складові компоненти, їх взаємозв’язки та поведінку.

Аналізуючи предметну галузь обліку періодичності технічного обслуговування телекомунікаційного обладнання можна виділити такі складові компоненти:

Сутності:

а) Обладнання (Equipment): Ця сутність описує обладнання, яке підлягає обслуговуванню. Вона має такі атрибути:

1) Id: Унікальний ідентифікатор обладнання;
2) Тип (Type): Тип обладнання;
3) Модель (Model): Модель обладнання;
4) Заводський номер (FactoryIdNumber): Заводський номер обладнання;
5) Інвентарний номер (Inventory Number): Інвентарний номер обладнання;
6) MAC-адреса (MACaddress): MAC-адреса обладнання;
7) IP-адреса (IpAddress): IP-адреса обладнання.

б) Технічне обслуговування (Maintenance): Ця сутність описує технічне обслуговування, яке виконується на обладнанні. Вона має такі атрибути:

1) Id: Унікальний ідентифікатор технічного обслуговування;
2) Назва технічного обслуговування (MaintenanceName): Назва технічного обслуговування;
3) Періодичність (Periodicity): Періодичність виконання технічного обслуговування.

в) Виконане технічне обслуговування (PerformedMaintenance): Ця сутність описує виконане технічне обслуговування. Вона має такі атрибути:

1) Id: Унікальний ідентифікатор виконаного технічного обслуговування;

2) Id технічного обслуговування (IdMaintenance): Ідентифікатор технічного обслуговування, яке було виконано;

3) Id обладнання (IdEquipment): Ідентифікатор обладнання, на якому було виконано технічне обслуговування;

4) Дата виконання (DateOfPerformed): Дата виконання технічного обслуговування;

5) Id відповідальної особи (IdResponsiblePerson): Ідентифікатор відповідальної особи за виконання технічного обслуговування;

6) Додаткова інформація (AdditionalInformation): Додаткова інформація про виконане технічне обслуговування.

г) Співробітник (Employee): Ця сутність описує співробітників, які виконують технічне обслуговування. Вона має такі атрибути:

1) Id: Унікальний ідентифікатор співробітника;

2) ПІБ (FullName): Прізвище, ім'я та по батькові співробітника;

3) Посада (Position): Посада співробітника.

Зв’язки:

а) Обладнання (Equipment) − Виконане технічне обслуговування (PerformedMaintenance): Один екземпляр обладнання може бути задіяний у багатьох виконаних технічних обслуговування.

б) Виконане технічне обслуговування (Performed Maintenance) - Співробітник (Employee): Одне технічне обслуговування може бути виконане одним співробітником.

Наступним кроком є створення візуального представлення предметної галузі за моделлю, тобто діаграми предметної галузі.

Діаграма предметної галузі створюється для візуального зображення ключових компонентів та представлення зв’язків між ними. Вона допомагає зробити складні системи більш зрозумілими та доступними для людей з різним рівнем технічної підготовки та спроектувати чіткі, ефективні, логічні системи.

За даною моделлю предметної галузі, діаграму представлено на рисунку 3.1.

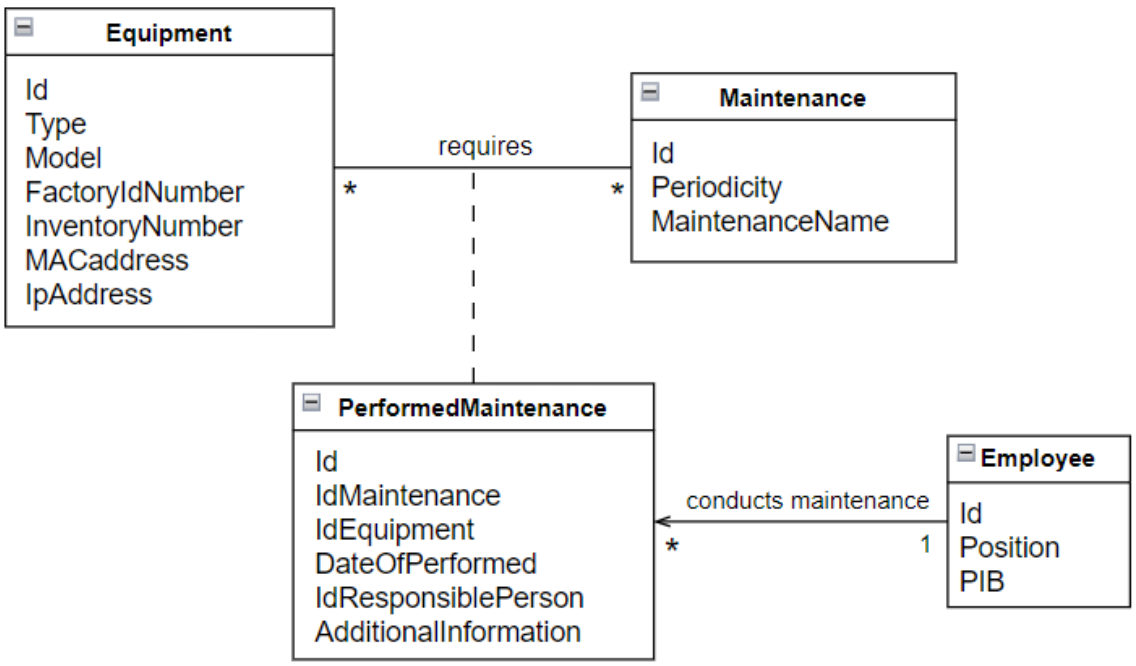


Рис. 3.1 Діаграма предметної галузі

3.2 Модель прецедентів

Модель прецедентів описує функціональні вимоги до системи. Вона ідентифікує її акторів та прецеденти. Актори – це зовнішні сутності, які взаємодіють із системою (люди, інші системи або програмні модулі). Прецеденти – це послідовності дій, які виконуються системою для досягнення певної мети актора.

Модель прецедентів системи застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання:

Актори:

- Працівник. Користувач системи, відповідальний за керування телекомунікаційним обладнанням та виконання завдань з технічного обслуговування.
- Адміністратор. Користувач системи з розширеними правами, який може керувати обладнанням, користувачами та записами про технічне обслуговування.

Система:

– Застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

Прецеденти:

– перегляд інформації про наявне обладнання. Розширення: пошук за критеріями;

– перегляд списку виконаного обслуговування. Розширення: вибір фільтрації та пошук та критеріями;

– додавання інформації про нове обслуговування;

– управління даними про види технічних обслуговувань. Розширення: створення, редагування, перегляд, видалення.

– управління даними про обладнання. Розширення: створення, редагування, перегляд, видалення.

– управління даними про працівників. Розширення: створення, редагування, перегляд, видалення.

Зв'язки:

- працівник може переглядати інформацію про наявне обладнання;
- працівник може переглядати список виконаного обслуговування;
- працівник може додавати інформацію про нове обслуговування;
- адміністратор може переглядати інформацію про наявне обладнання;
- адміністратор може переглядати список виконаного обслуговування;
- адміністратор може додавати інформацію про нове обслуговування;
- адміністратор може керувати даними про Т.О., обладнання, користувачів.

На рисунку 3.2 зображено діаграму прецедентів, яка візуально відображає взаємодію між двома акторами та системою.

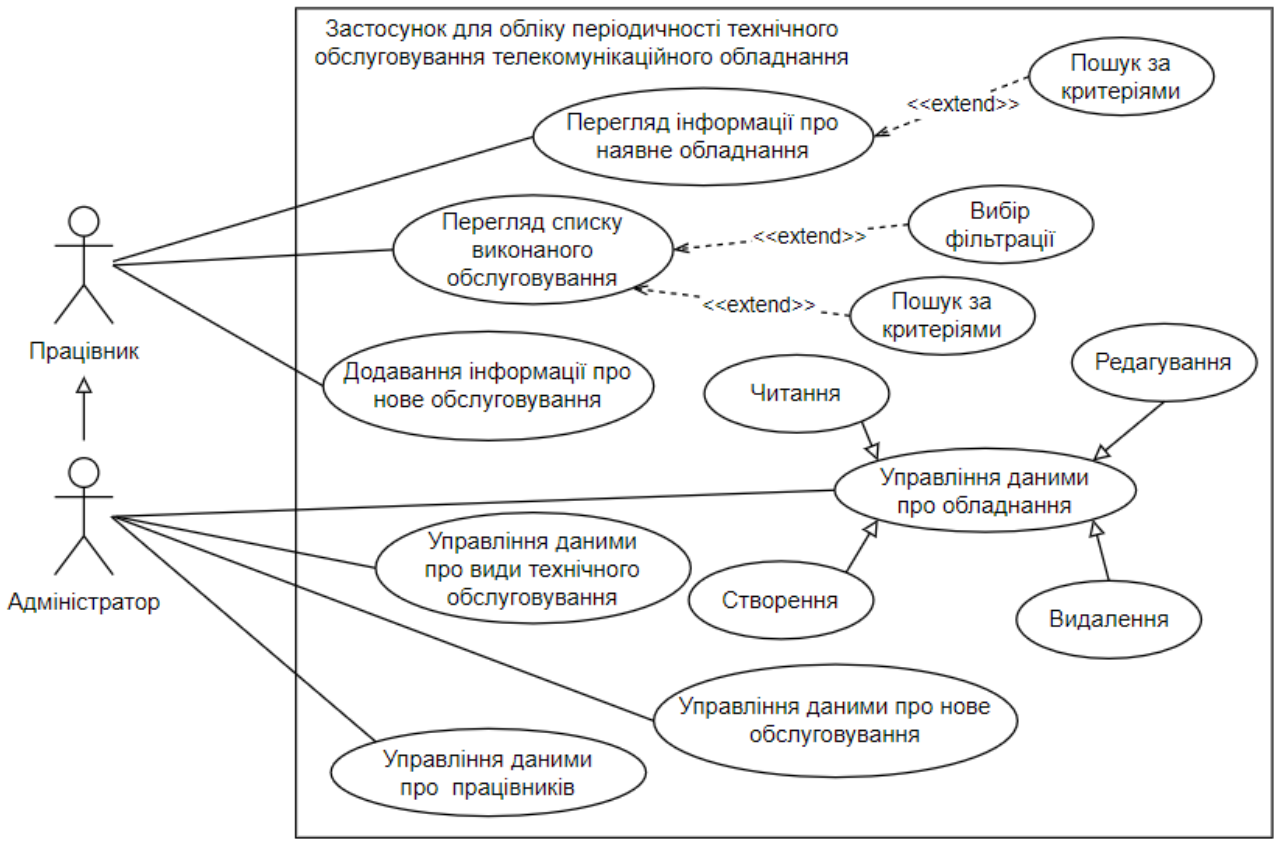


Рис. 3.2 Діаграма прецедентів

3.4 ER-модель бази даних

ER-модель (також відома, як модель сутність-зв’язок) є концептуальною моделлю для опису структури бази даних. За допомогою неї можна виділити основні сутності, атрибути та зв’язки між ними. Сутності представляють собою реальні об’єкти або абстрактні явища, які мають певне значення для предметної галузі. Кожна сутність має певний набір атрибутів та унікальну назву. Атрибути описують властивості сутності. Зв’язки позначають взаємозв’язки між сутностями.

ER-модель слугує основою для подальшого проектування та реалізації бази даних. Вона надає чітке уявлення про те, як дані організовані та пов’язані між собою.

Компоненти ER-моделі бази даних застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання:

Сутності:

– обладнання (Equipments). Атрибути: тип обладнання, модель, заводський номер, інвентарний номер, MAC-адреса, IP-адреса;

– працівники (Employees). Атрибути: посада, ПІБ (прізвище, ім’я, по батькові);

– проведені технічні обслуговування (Performed Maintenances). Атрибути: Id-номер технічного обслуговування, Id-номер обладнання, Id-номер відповідального працівника, дата проведення, додаткова інформація;

– технічні обслуговування (Maintenances). Атрибути: періодичність, назва технічного обслуговування.

Зв’язки:

– одна одиниця обладнання може проходити багато технічних обслуговувань (один до багатьох);

– один тип технічних обслуговувань може виконуватись над багатьма одиницями обладнання (один до багатьох);

– один працівник може виконувати багато технічних обслуговувань (один до багатьох).

Для візуального зображення ER-моделі використовують ER-діаграми. ER-діаграму бази даних застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання зображено на рисунку 3.3.

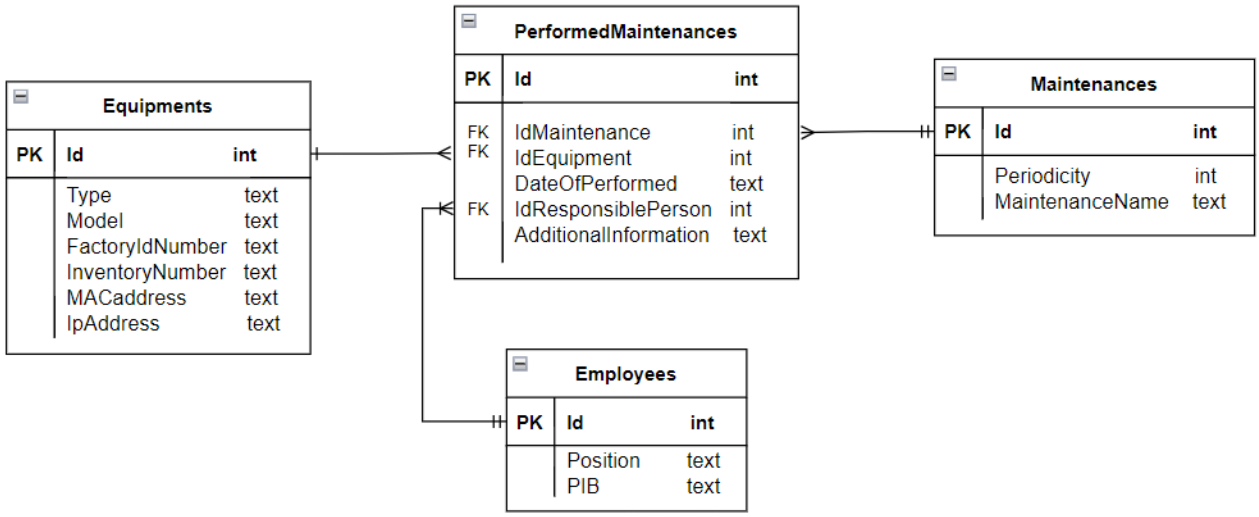


Рис. 3.3 ER- діаграма

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Підключення до бази даних

Для підключення та подальшої роботи з базою даних, в першу чергу, було підключено NuGet-пакети Microsoft.EntityFrameworkCore.Sqlite та Microsoft.EntityFrameworkCore.Tools.

Підключення до бази даних відбувається паралельно зі створенням моделей за допомогою підходу Database first. Такий підхід дозволяє, за допомогою EF Core, на основі вже створеної бази даних автоматично створити класи сутностей та клас контексту бази даних.

У консолі диспетчера пакетів було прописано команду: Scaffold-DbContext" Data Source=.\DataBase\MaoTEDB.db" Microsoft.EntityFrameworkCore.Sqlite -OutputDir Models. Де Data Source=.\DataBase\MaoTEDB.db – шлях до створеної бази даних застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання, OutputDir Models – команда для створення моделей (класів сутностей) і класу контексту.

На рисунку 4.1 зображено код підключення до бази даних у класі контексту бази даних.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
=> optionsBuilder.UseSqlite("Data Source=.\\DataBase\\MaoTEDB.db");
```

Рис. 4.1 Підключення до бази даних у класі контексту

4.2 Реалізація сутностей Entity Framework Core

У Entity Framework Core сутності бази даних реалізуються як класи, для відображення даних в об’єктній моделі. Клас сутності (модель) містить властивості відповідно до атрибутів сутності бази даних. Також для реалізації зв’язків та функцій роботи з даними прописується клас контексту, він дозволяє

виконувати операції CRUD (створення, перегляд, оновлення, видалення) з даними у базі.

```
public partial class Employee
{
    Ссылок: 4
    public int Id { get; set; }

    Ссылок: 3
    public string Position { get; set; } = null!;

    Ссылок: 5
    public string FullName { get; set; } = null!;

    Ссылок: 1
    public virtual ICollection<PerformedMaintenance> PerformedMaintenances { get; set; } = new List<PerformedMaintenance>();
}
```

Рис. 4.2 Клас сутності «Працівник»

```
public partial class Equipment
{

    public int Id { get; set; }


    public string Type { get; set; } = null!;


    public string Model { get; set; } = null!;

    Ссылок: 4
    public string FactoryIdNumber { get; set; } = null!;

    Ссылок: 12
    public string InventoryNumber { get; set; } = null!;

    Ссылок: 4
    public string? Macaddress { get; set; }

    Ссылок: 4
    public string? IpAddress { get; set; }

    Ссылок: 1
    public virtual ICollection<PerformedMaintenance> PerformedMaintenances { get; set; } = new List<PerformedMaintenance>();
}
```

Рис. 4.3 Клас сутності «Обладнання»

```
public partial class Maintenance
{
    Ссылок: 9
    public int Id { get; set; }

    Ссылок: 3
    public int Periodicity { get; set; }

    Ссылок: 13
    public string MaintenanceName { get; set; } = null!;

    Ссылок: 1
    public virtual ICollection<PerformedMaintenance> PerformedMaintenances { get; set; } = new List<PerformedMaintenance>();

}
```

Рис. 4.4 Клас сутності «Технічне обслуговування»

```
public partial class PerfermedMaintenance
{
    Ссылок: 3
    public int Id { get; set; }

    Ссылок: 9
    public int IdMaintenance { get; set; }

    Ссылок: 9
    public int IdEquipment { get; set; }

    Ссылок: 9
    public string DateOfPerfermed { get; set; } = null!;

    Ссылок: 3
    public int IdEmployee { get; set; }

    Ссылок: 3
    public string? AdditionalInformation { get; set; }

    Ссылок: 1
    public virtual Employee IdEmployeeNavigation { get; set; } = null!;
    Ссылок: 1
    public virtual Equipment IdEquipmentNavigation { get; set; } = null!;
    Ссылок: 1
    public virtual Maintenance IdMaintenanceNavigation { get; set; } = null!;
}
```

Рис. 4.5 Клас сутності «Виконане технічне обслуговування»

```
public partial class MaoTedbContext : DbContext
    Ссылок: 36
{   public MaoTedbContext()...
    Ссылок: 0
    public MaoTedbContext(DbContextOptions<MaoTedbContext> options)...
    Ссылок: 11
    public virtual DbSet<Employee> Employees { get; set; }
    Ссылок: 22
    public virtual DbSet<Equipment> Equipments { get; set; }
    Ссылок: 17
    public virtual DbSet<Maintenance> Maintenances { get; set; }
    Ссылок: 14
    public virtual DbSet<PerfermedMaintenance> PerfermedMaintenances { get; set; }
    Ссылок: 0
    public View.EquipmentForm EquipmentForm...
    Ссылок: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder.UseSqlite("Data Source=.\\DataBase\\MaoTEDB.db");
    Ссылок: 0
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Employee>(entity =>...);
        modelBuilder.Entity<Equipment>(entity =>...);
        modelBuilder.Entity<Maintenance>(entity =>...);
        modelBuilder.Entity<PerfermedMaintenance>(entity =>...);
        OnModelCreatingPartial(modelBuilder);
    }
    Ссылок: 1
    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

Рис. 4.6 Клас контексту бази даних

4.3 Реалізація функціональних вимог

Функції додавання нового обладнання, працівників, технічних обслуговувань, проведених технічних обслуговуваннях реалізовано за допомогою таких методів: NewEquipment(), NewEmployee(), NewMaintenance(),

NewPerfermedMaintenance(), відповідно. Програмний код цих методів наведено на рисунках 4.7-10.

```
public static void NewEquipment(string newType, string newModel, string newFactoryNum, string newInvNum, string newMac, string newIp)
{
    using (var db = new MaoTedbContext())
    {
        var std = new Equipment { Type = newType, Model = newModel, FactoryIdNumber = newFactoryNum,
                                  InventoryNumber = newInvNum, Macaddress = newMac, IpAddress = newIp};
        db.Equipments.Add(std);
        db.SaveChanges();
    }
}
```

Рис. 4.7 Код методу NewEquipment()

```
public static void NewMaintenance(int newPeriodicity, string newMaintenanceName)
{
    using (var db = new MaoTedbContext())
    {
        var std = new Maintenance { Periodicity = newPeriodicity, MaintenanceName = newMaintenanceName };
        db.Maintenances.Add(std);
        db.SaveChanges();
    }
}
```

Рис. 4.8 Код методу NewMaintenance ()

```
public static void NewEmployee(string newPosition, string newFullName)
{
    using (var db = new MaoTedbContext())
    {
        var std = new Employee { Position = newPosition, FullName = newFullName };
        db.Employees.Add(std);
        db.SaveChanges();
    }
}
```

Рис. 4.9 Код методу NewEmployee ()

```
public void NewPerfermedMaintenance(int idMaintenanceNew, int idEquipmentNew, string dateOfperformedNew, int idEmployeeNew, string additionalInformationNew)
{
    using (var db = new MaoTedbContext())
    {
        var std = new PerfermedMaintenance
        {
            IdMaintenance = idMaintenanceNew, IdEquipment = idEquipmentNew, DateOfPerfermed = dateOfperformedNew,
            IdEmployee = idEmployeeNew,AdditionalInformation = additionalInformationNew
        };
        db.PerfermedMaintenances.Add(std);
        db.SaveChanges();
    }
}
```

Рис. 4.10 Код методу NewPerfermedMaintenance ()

Функції редагування даних про існуюче обладнання, працівників, технічні обслуговування, проведене технічне обслуговування реалізовано за допомогою

таких методів: UpdEquipment(), UpdEmployee(), UpdMaintenance(), UpdPerfermedMaintenance(), відповідно. Програмний код цих методів наведено на рисунках 4.11-14.

```
public static void UpdEmployee(int id, string newPosition, string newFullName)
{
    using (var db = new MaoTedbContext())
    {
        var std = db.Employees.FirstOrDefault(e => e.Id == id);
        std.Position = newPosition;
        std.FullName = newFullName;
        db.SaveChanges();
    }
}
```

Рис. 4.11 Код методу UpdEmployee ()

```
public static void UpdPerfermedMaintenance(int id, int idMaintenanceNew, int idEquipmentNew, string dateOfperformedNew,
                                           int idEmployeeNew, string additionalInformationNew)
{
    using (var db = new MaoTedbContext())
    {
        var std = db.PerfermedMaintenances.FirstOrDefault(e => e.Id == id);
        std.DateOfPerfermed = dateOfperformedNew;
        std.IdMaintenance = idMaintenanceNew;
        std.IdEquipment = idEquipmentNew;
        std.AdditionalInformation = additionalInformationNew;
        std.IdEmployee = idEmployeeNew;
        db.SaveChanges();
    }
}
```

Рис. 4.12 Код методу UpdPerfermedMaintenance ()

```
public static void UpdMaintenance(int id, int newPeriodicity, string newMaintenanceName)
{
    using (var db = new MaoTedbContext())
    {
        var std = db.Maintenances.FirstOrDefault(e => e.Id == id);
        std.MaintenanceName = newMaintenanceName;
        std.Periodicity = newPeriodicity;
        db.SaveChanges();
    }
}
```

Рис. 4.13 Код методу UpdMaintenance ()

```
public static void UpdEquipment(int id, string newType, string newModel, string newFactoryNum, string newInvNum, string newMac, string newIp)
{
    using (var db = new MaoTedbContext())
    {
        var std = db.Equipments.FirstOrDefault(e => e.Id == id);
        std.Type = newType;
        std.Model = newModel;
        std.Macaddress = newMac;
        std.IpAddress = newIp;
        std.FactoryIdNumber = newFactoryNum;
        std.InventoryNumber = newInvNum;
        db.SaveChanges();

    }
}
```

Рис. 4.14 Код методу UpdEquipment ()

Видалення обраної одиниці обладнання, працівників, технічного обслуговування, проведеного технічного обслуговування відбувається шляхом вибору потрібної таблиці з випадаючого списку, натисканням на потрібному рядку та натисканням на кнопку «Видалити». Далі висвічується повідомлення про підтвердження операції, в якому треба натиснути клавішу «Ок» і обрана одиниця буде видалена з бази даних. Програмно це реалізовано кодом, який зображено на рисунку 4.15.

```
private void btnDelete_Click(object sender, EventArgs e)
{
    MaoTedbContext db = new MaoTedbContext();
    DataGridViewRow selectedRow = dgvMain.CurrentRow;
    string selectedId = selectedRow.Cells["Id"].Value.ToString();
    int idFromDgv = Int32.Parse(selectedId);
    DialogResult dialogResult = MessageBox.Show("Видалити рядок з Id: " + idFromDgv, "Повідомлення", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        switch (cbTablesName.Text)
        {
            case "Проведене ТО":
                var selectedPerfMaintenance = db.PerfermedMaintenances.FirstOrDefault(pm => pm.Id == idFromDgv);
                db.PerfermedMaintenances.Remove(selectedPerfMaintenance);
                db.SaveChanges();
                dgvMain.DataSource = db.PerfermedMaintenances.ToList();
                break;
            case "Обладнання":
                var selectedEquipment = db.Equipments.FirstOrDefault(eq => eq.Id == idFromDgv);
                db.Equipments.Remove(selectedEquipment);
                db.SaveChanges();
                dgvMain.DataSource = db.Equipments.ToList();
                break;
            case "Працівники":
                var selectedEmployee = db.Employees.FirstOrDefault(em => em.Id == idFromDgv);
                db.Employees.Remove(selectedEmployee);
                db.SaveChanges();
                dgvMain.DataSource = db.Employees.ToList();
                break;
            case "ТО":
                var selectedMaintenance = db.Maintenances.FirstOrDefault(mn => mn.Id == idFromDgv);
                db.Maintenances.Remove(selectedMaintenance);
                db.SaveChanges();
                dgvMain.DataSource = db.Maintenances.ToList();
                break;
            default:
                MessageBox.Show("Оберіть таблицю");
                break;
        }
    }
}
```

Рис. 4.15 Код функції видалення

Пошук необхідної одиниці телекомунікаційного обладнання за інвентарним номером і моделлю та пошук виконаного технічного обслуговування за датою проведення реалізовано через ввід необхідних даних у текстове поле на натискання клавіші «Enter», після чого виводяться тільки відповідні до пошуку дані. Код цих функцій представлено на рисунках 4.16-18.

```
private void tbInvSearch_KeyDown(object sender, KeyEventArgs e)
{
    MaoTedbContext db = new MaoTedbContext();
    if (e.KeyCode == Keys.Enter)
    {
        e.SuppressKeyPress = true;

        dgvEquipment.DataSource = (from eq in db.Equipments
                                   where eq.InventoryNumber == tbInvSearch.Text
                                   select new { eq.Type, eq.Model, eq.InventoryNumber }).ToList();
    }
    else
    {
        dgvEquipment.DataSource = (from eq in db.Equipments
                                   select new { eq.Type, eq.Model, eq.InventoryNumber }).ToList();
    }
}
```

Рис. 4.16 Код пошуку необхідної одиниці телекомунікаційного обладнання за інвентарним номером

```
private void tbModelSearch_KeyDown(object sender, KeyEventArgs e)
{
    MaoTedbContext db = new MaoTedbContext();
    if (e.KeyCode == Keys.Enter)
    {
        e.SuppressKeyPress = true;
        dgvEquipment.DataSource = (from eq in db.Equipments
                                   where eq.Model == tbModelSearch.Text
                                   select new { eq.Type, eq.Model, eq.InventoryNumber }).ToList();
    }
    else
    {
        dgvEquipment.DataSource = (from eq in db.Equipments
                                   select new { eq.Type, eq.Model, eq.InventoryNumber }).ToList();
    }
}
```

Рис. 4.17 Код пошуку необхідної одиниці телекомунікаційного обладнання за моделлю

```
private void tbInvSearchDate_KeyDown(object sender, KeyEventArgs e)
{
    MaoTedbContext db = new MaoTedbContext();
    if (e.KeyCode == Keys.Enter)
    {
        e.SuppressKeyPress = true;

        dgvPerfMain.DataSource = (from pm in db.PerfermedMaintenances
                                  join m in db.Maintenances on pm.IdMaintenance equals m.Id
                                  join eq in db.Equipments on pm.IdEquipment equals eq.Id
                                  where pm.DateOfPerfermed == tbInvSearchDate.Text
                                  select new { eq.Type, eq.Model, m.MaintenanceName, pm.DateOfPerfermed }
                                 ).ToList();
    }
    else
    {
        dgvPerfMain.DataSource = (from pm in db.PerfermedMaintenances
                                  join m in db.Maintenances on pm.IdMaintenance equals m.Id
                                  join eq in db.Equipments on pm.IdEquipment equals eq.Id
                                  select new { eq.Type, eq.Model, m.MaintenanceName, pm.DateOfPerfermed }).ToList();
    }
}
```

Рис. 4.18 Код пошуку виконаного технічного обслуговування за датою проведення

Перегляд повної інформації про обрану одиницю телекомунікаційного обладнання відбувається після подвійного кліку на рядку з необхідною одиницею, після чого висвічується повідомлення, яке містить усю інформацію про цю одиницю. Код програмної реалізації цієї функції зображено на рисунку 4.19.

```
private void dgvEquipment_DoubleClick(object sender, EventArgs e)
{
    MaoTedbContext db = new MaoTedbContext();
    DataGridViewRow selectedRow = dgvEquipment.CurrentRow;
    string selectedInventoryNumber = selectedRow.Cells["InventoryNumber"].Value.ToString();
    var selectedEquipment = db.Equipments.FirstOrDefault(eq => eq.InventoryNumber == selectedInventoryNumber);

    if (selectedEquipment != null)
    {
        string message = $"Інвентарний номер: {selectedEquipment.InventoryNumber}\n" +
                         $"Тип обладнання: {selectedEquipment.Type}\n" +
                         $"Модель: {selectedEquipment.Model}\n" +
                         $"Заводський номер: {selectedEquipment.FactoryIdNumber}\n" +
                         $"MAC-адрес: {selectedEquipment.Macaddress ?? "N/A"}\n" +
                         $"IP-адрес: {selectedEquipment.IpAddress ?? "N/A"}";

        MessageBox.Show(message, "Повна інформація про обладнання");
    }
}
```

Рис. 4.19 Код функції перегляду повної інформації про обрану одиницю телекомунікаційного обладнання

4.4 Проектування архітектури

Для проектування архітектури, візуалізації структури системи, визначення залежностей використовують діаграму пакетів. Пакетами називають логічне групування елементів моделі, наприклад, класів, інтерфейсів, компонентів.

На рисунку 4.20 представлено діаграму пакетів застосунку з обліку періодичності технічного обслуговування телекомунікаційного обладнання. Презентаційний шар (Presentation Layer) відповідає за взаємодію з користувачами і містить форми, які відображаються на екрані. Шар бізнес-логіки (Business Logic Layer) містить бізнес-логіку, яка обробляє дані, отримані з Presentation Layer, та взаємодіє з Data Access Layer для доступу до даних. Шар доступу до даних (Data Access Layer) забезпечує доступ до даних.

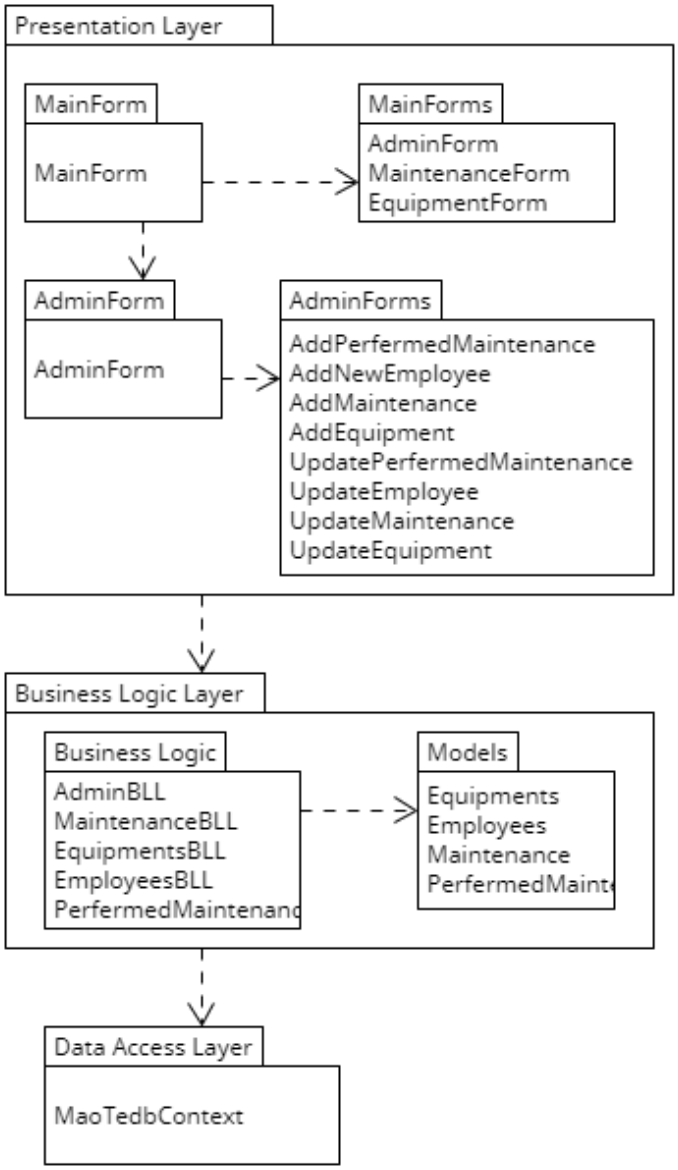


Рис. 4.20 Архітектура застосунку. Діаграма пакетів

4.5 Діаграма послідовності

Діаграма послідовності використовується для моделювання динамічних аспектів системи. Вона показує, як об'єкти взаємодіють між собою протягом часу, вказуючи на порядок виконання повідомлень між ними. Діаграма послідовності використовується для відображення послідовності обміну повідомленнями, які призводять до виконання певного сценарію в системі. Її основними компонентами є актори, об'єкти, лінії життя, повідомлення, активності.

На рисунку 4.21 представлено діаграму послідовності прецедента «Додавання інформації про нове обслуговування» на рівні Presentation Layer. Вона відображає сценарій обробки кліка на кнопку btnSaveAddPerMnt_Click у формі AddPerfermedMaintenance. Показує, як цей клік ініціює виклик методу, а також подальшу логіку обробки та взаємодії між класами AddPerfermedMaintenance, AddPerfermedMaintenanceBLL і MessageBox.

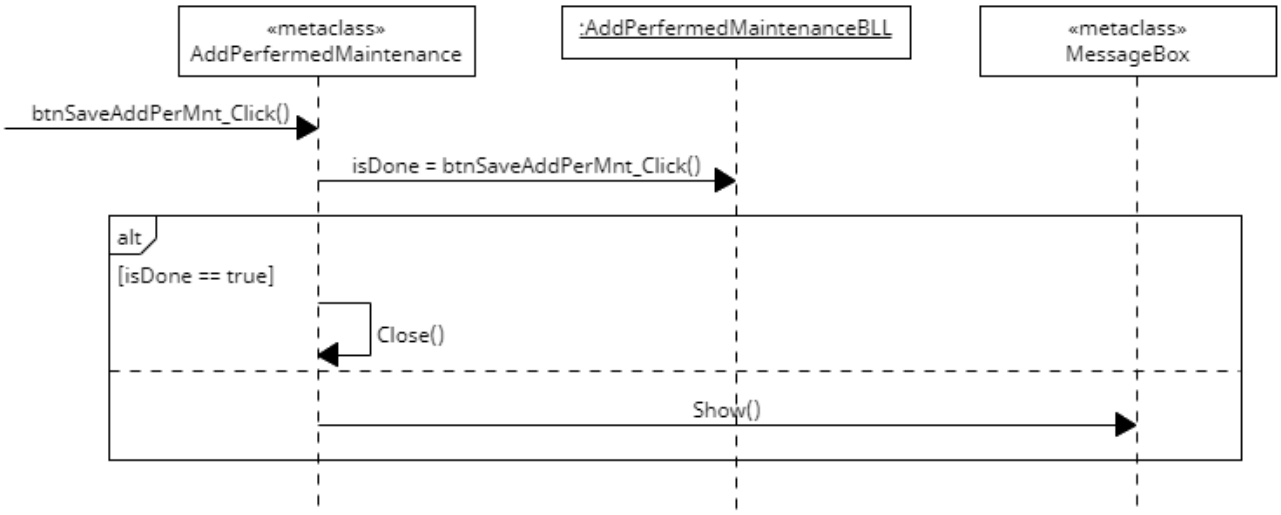


Рис 4.21 Presentation Layer

Опис процесу:

1. Користувач натискає кнопку "Зберегти".
2. Клас AddPerfermedMaintenanceBLL виконує код для додавання виконаного технічного обслуговування.

3. Клас AddPerfermedMaintenanceBLL надсилає повідомлення isDone(), щоб перевірити, чи було успішно додано виконане технічне обслуговування.

4. Якщо виконане технічне обслуговування було успішно додано, клас MessageBox надсилає повідомлення Close(), щоб закрити діалогове вікно повідомлення.

5. Якщо виконане технічне обслуговування не було успішно додано, клас MessageBox надсилає повідомлення Show(), щоб відобразити діалогове вікно повідомлення з повідомленням про помилку.

На рисунку 4.22 представлено діаграму послідовності прецедента «Додавання інформації про нове обслуговування» на рівні Business Logic Layer. Вона демонструє процес виконання методу btnSaveAddPerMnt_Click у класі AddPerformedMaintenanceBLL. Описує перевірку введених даних, вибір значень з бази даних та додавання нового запису про виконане обслуговування.

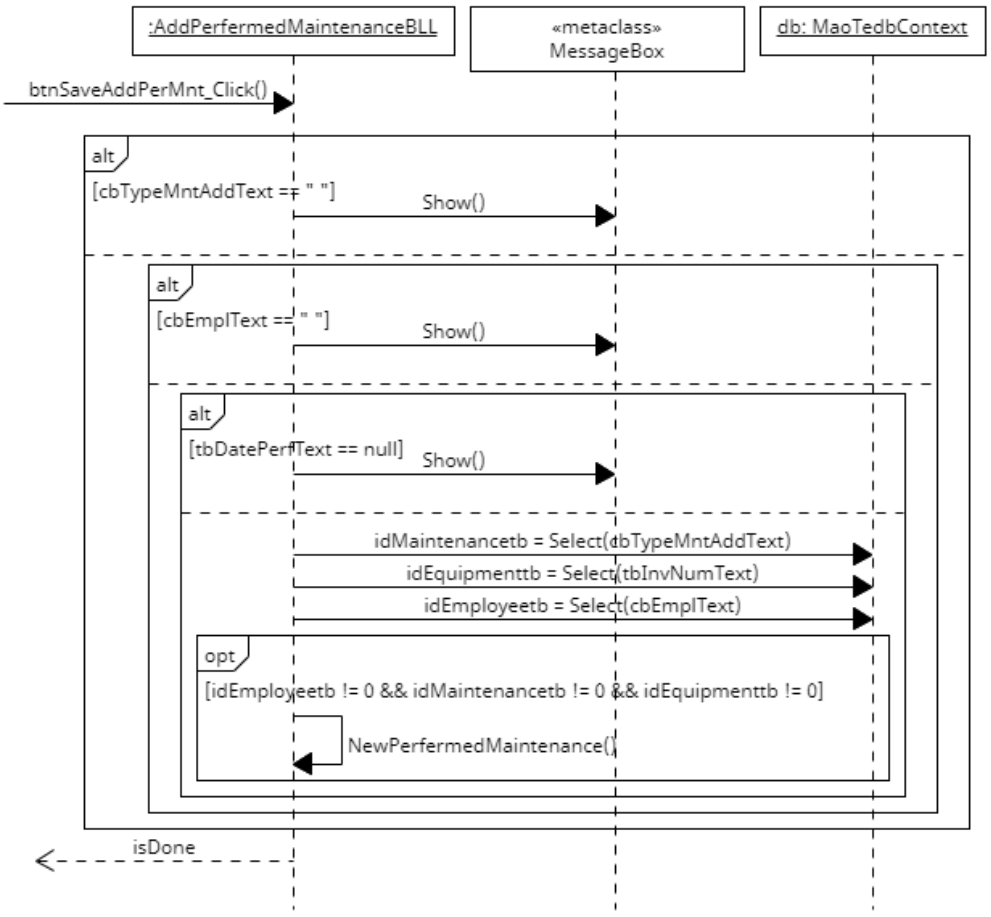


Рис 4.22 Business Logic Layer

4.6 Діаграма класів

Діаграма класів використовується для візуалізації об’єктно-орієнтованих систем. Для опису структури системи, вона демонструє її властивості, класи, операції та зв’язки між об’єктами.

Елементами діаграми класів є назва класу, атрибути класу, методи та зв’язки. Між об’єктами на діаграмі класів можуть бути такі зв’язки: агрегація (коли один клас є частиною іншого), композиція (один клас повність володіє іншим та контролює його життєвий цикл), залежність (один клас використовує функціонал іншого тимчасово), спадкування (один клас успадковує властивості та методи іншого класу).

На рисунках 4.23 та 4.24 наведено діаграми класів виконаних обслуговувань та працівників.

Зв’язки залежності на діаграмі виконаних обслуговувань: клас MainForm використовує клас AdminForm, клас AdminForm використовує клас AddPerfermedMaintenance, клас AddPerfermedMaintenance використовує клас PerfermedMaintenanceBLL, клас PerfermedMaintenanceBLL використовує клас PerfermedMaintenance.

Зв’язки залежності на діаграмі працівників: клас MainForm використовує клас AdminForm, клас AdminForm використовує класи AddNewEmployee та UpdateEmployee, класи AddNewEmployee та UpdateEmployee використовують клас EmployeeBLL, клас EmployeeBLL використовує клас Employee.

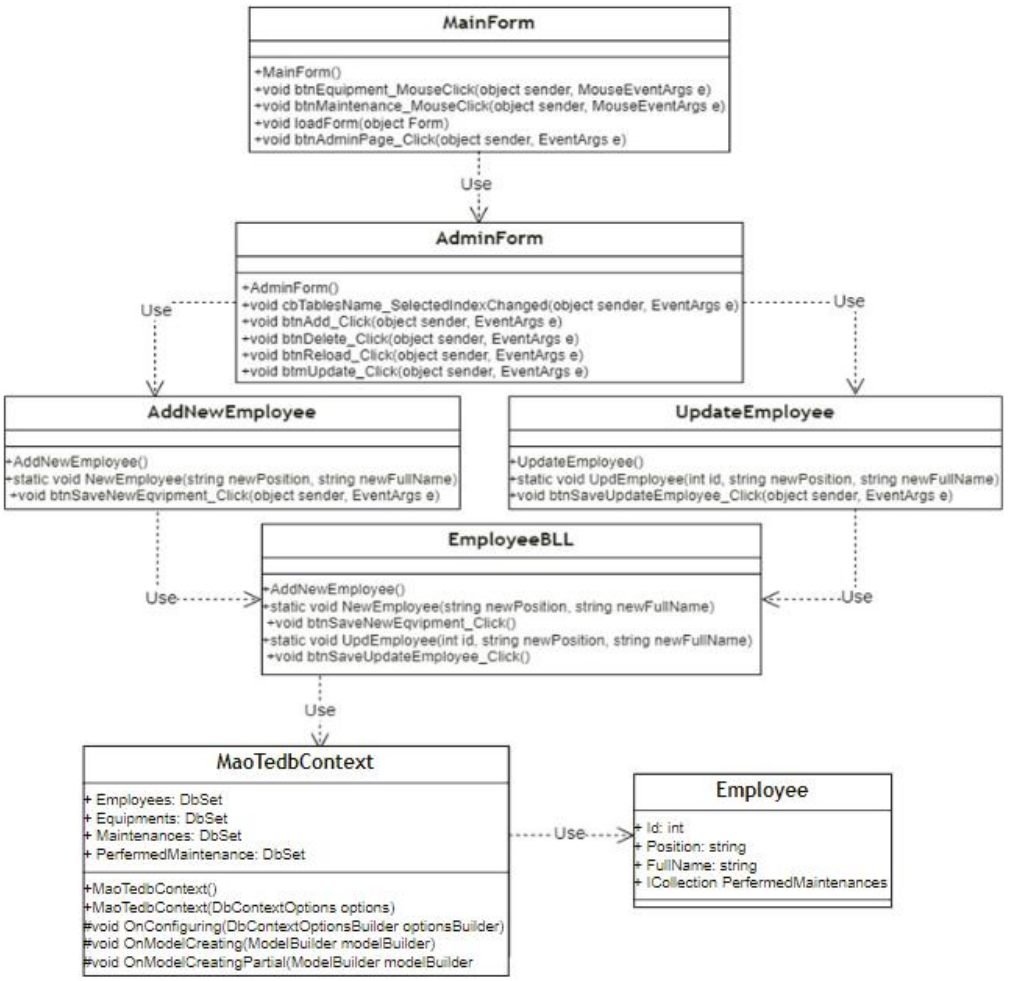


Рис. 4.23 Діаграма класів працівників

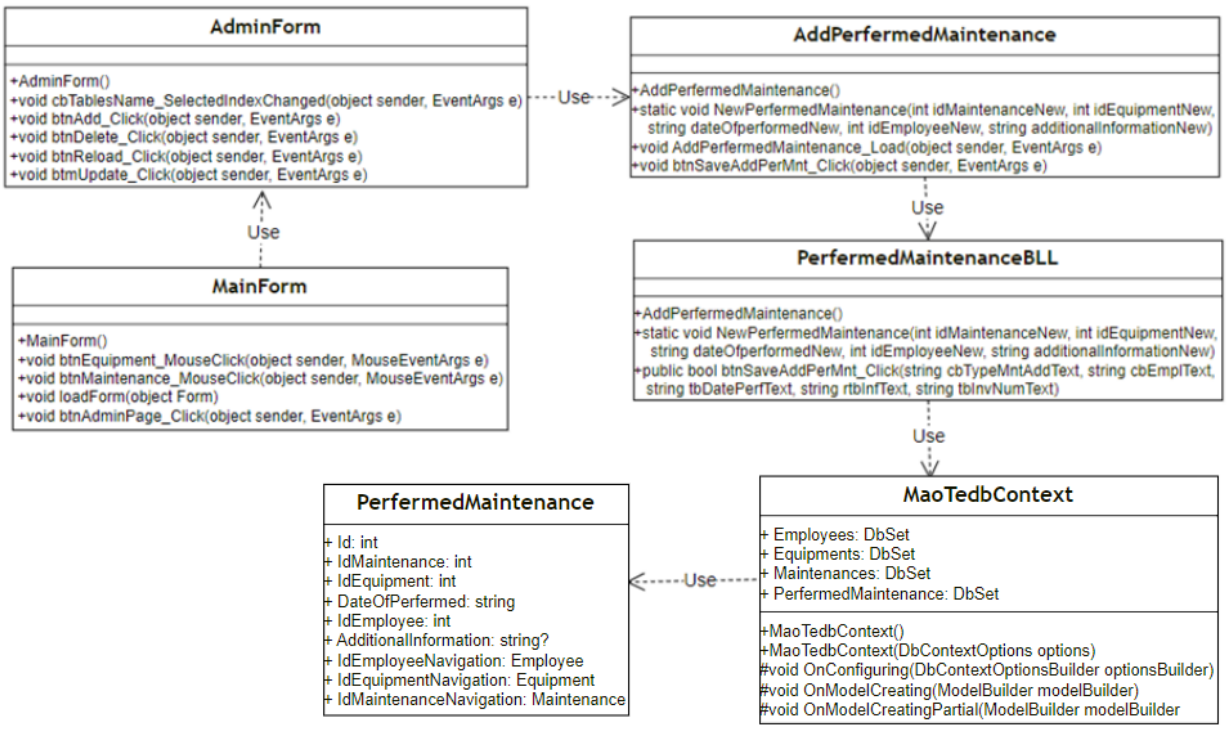


Рис. 4.24 Діаграма класів виконаних обслуговувань

5 ТЕСТУВАННЯ ЗАСТОСУНКУ

Для застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання було проведено тестування функціональності, з метою перевірки чи виконує розроблений застосунок усі функціональні вимоги.

Результати тестування представлено у таблиці 5.1.

Таблиця 5.1

Тестування функціональності застосунку

Вікно яке тестувалось	Протестована функція	Очікуваний результат	Результат тесту
Вікно додавання нового обладнання	Додавання нового обладнання	Після введення даних про обладнання у відповідні поля та натискання кнопки «Зберегти», у базу даних додалось нове обладнання	Успіх
Вікно додавання нового технічного обслуговування	Додавання нового технічного обслуговування	Після введення даних про технічне обслуговування у відповідні поля та натискання кнопки «Зберегти», у базу даних додалось нове технічне обслуговування	Успіх
Вікно додавання нового працівника	Додавання нового працівника	Після введення даних про працівника у відповідні поля та натискання кнопки «Зберегти», у базу даних додався новий працівник	Успіх

Продовження таблиці 5.1

Тестування функціональності застосунку

Вікно яке тестувалось	Протестована функція	Очікуваний результат	Результат тесту
Вікно додавання нового типу технічного обслуговування	Додавання нового типу технічного обслуговування	Після введення даних про тип технічного обслуговування у відповідні поля та натискання кнопки «Зберегти», у базу даних додався новий тип технічного обслуговування	Успіх
Вікно перегляду телекомунікаційного обладнання	Перегляд усього телекомунікаційного обладнання	У вікні відображається усе наявне телекомунікаційне обладнання	Успіх
Вікно перегляду виконаних обслуговувань	Перегляд усіх виконаних обслуговувань	У вікні відображається усі виконані обслуговування	Успіх
Вікно перегляду телекомунікаційного обладнання	Перегляд повної інформації про обрану одиницю обладнання	Після подвійного кліку по рядку з обраною одиницею обладнання з’являється повідомлення з повною інформацією	Успіх
Вікно перегляду телекомунікаційного обладнання	Пошук необхідної одиниці обладнання за інвентарним номером або моделлю	Після вводу інвентарного номеру або моделі у відповідні поля та натискання клавіші «Enter» відображаються одиниці обладнання за якими закріплений даний номер або модель	Успіх

Продовження таблиці 5.1

Тестування функціональності застосунку

Вікно яке тестувалось	Протестована функція	Очікуваний результат	Результат тесту
Вікно перегляду виконаних обслуговувань	Пошук виконаного технічного обслуговування за датою проведення	Після вводу дати у відповідне поля та натискання клавіші «Enter» відображаються проведені технічні обслуговування за цю дату	Успіх
Вікно перегляду виконаних обслуговувань	Фільтрування за типом обладнання або видом технічного обслуговування, при перегляді виконаних технічних обслуговувань	Після вибору типу обладнання або виду технічного обслуговування з відповідного випадаючого списку відображаються виконані технічні обслуговування для цього типу обладнання чи виду технічного обслуговування	Успіх
Вікно видалення обладнання	Видалення обладнання	Після вибору рядку з обладнанням, яке необхідно видалити, натиснення кнопки «Видалити» та кнопки «Ок» обрана одиниця видаляється з бази	Успіх
Вікно видалення технічного обслуговування	Видалення технічного обслуговування	Після вибору рядку з технічним обслуговуванням, натиснення кнопки «Видалити» та кнопки «Ок» обрана одиниця видаляється з бази	Успіх

Продовження таблиці 5.1

Тестування функціональності застосунку

Вікно яке тестувалось	Протестована функція	Очікуваний результат	Результат тесту
Вікно видалення працівника	Видалення працівника	Після вибору рядку з працівником, якого необхідно видалити, натиснення кнопки «Видалити» та кнопки «Ок» у повідомленні для підтвердження обрана одиниця видаляється з бази	Успіх
Вікно видалення типу технічного обслуговування	Видалення типу технічного обслуговування	Після вибору рядку з типом технічного обслуговування, яке необхідно видалити, натиснення кнопки «Видалити» та кнопки «Ок» у повідомленні для підтвердження обрана одиниця видаляється з бази	Успіх
Вікно редагування обладнання	Редагування обладнання	Після редагування даних про обладнання у відповідних полях та натискання кнопки «Зберегти», у базі даних оновлюються дані про обране обладнання	Успіх
Вікно редагування технічного обслуговування	Редагування технічного обслуговування	Після редагування даних про технічне обслуговування у відповідних полях та натискання кнопки «Зберегти», у базі даних оновлюються дані про обране технічне обслуговування	Успіх

Продовження таблиці 5.1

Тестування функціональності застосунку

Вікно яке тестувалось	Протестована функція	Очікуваний результат	Результат тесту
Вікно редагування працівника	Редагування працівника	Після редагування даних про працівника у відповідних полях та натискання кнопки «Зберегти», у базі даних оновлюються дані	Успіх
Вікно редагування технічного обслуговування	Редагування типу технічного обслуговування	Після редагування даних про тип технічного обслуговування у відповідних полях та натискання кнопки «Зберегти», у базі даних оновлюються дані про обраний тип технічного обслуговування	Успіх

ВИСНОВКИ

1. Було проведено аналіз предметної області, виявлено основні проблеми такі, як: системні адміністратори можуть нести відповідальність за тисячі одиниць телекомунікаційного обладнання, що може ускладнити відстеження та виявлення потенційних проблем; забезпечення безперервної та безперебійної роботи мережевого та серверного обладнання, у деяких випадках системним адміністраторам може знадобитися працювати в ізольованих мережах, де вони не мають доступу до Інтернету або інших зовнішніх ресурсів.

2. Проведено аналіз інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання. На основі отриманих результатів було сформовано функціональні та нефункціональні вимоги.

3. Створено діаграму прецедентів на основі моделі прецедентів для візуального зображення взаємодії користувачів із системою.

4. Розроблено застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання мовою C#. За допомогою середовища розробки Visual Studio та платформи .NET Framework. Для створення інтерфейсу користувача було використано графічну підсистему WinForms. В якості системи керування базою даних було обрано SQLite та EntityFramework Core для зваємодії з нею.

5. Проведено функціональне тестування з метою перевірки застосунку на відповідність визначеним функціональним вимогам. Результат тестування успішний, що показує працездатність розробленого застосунку.

Кваліфікаційна робота пройшла апробацію на наступних конференціях:

1. Пермякова О.О., Гаманюк І.М. Використання шаблону MVVM в застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2024 р., Київ,

Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.54-56.

2. Пермякова О.О., Гаманюк І.М. Переваги використання СКБД SQLITE в застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.117-119.

ПЕРЕЛІК ПОСИЛАНЬ

1. C Sharp [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp.
2. Entity Framework Core [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-gb/ef/core/.
3. HippoCMMS [Електронний ресурс] – Режим доступу до ресурсу: https://eptura.com/hippocmms/.
4. Microsoft Excel [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/excel.
5. Overview of .NET Framework [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview.
6. What is Visual Studio? [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-gb/visualstudio/get-started/visual-studio-ide?view=vs-2022.
7. Windows Forms overview [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8.
8. Windows Notepad [Електронний ресурс] – Режим доступу до ресурсу: https://apps.microsoft.com/detail/9msmlrh6lzf3?hl=en-gb&gl=UA.
9. Експлуатація телекомунікаційних систем / Г. Ф.Конахович, О. П. Ткаліч, В. М. Чуприн, І. О. Мачалін. – Київ, 2014. – 800 с.
10. Inspection and Cleaning Procedures for Fiber-Optic Connections [Електронний ресурс] – Режим доступу до ресурсу: https://www.cisco.com/c/en/us/support/docs/optical/synchronous-digital-hierarchy-sdh/51834-cleanfiber2.html?referring_site=bodynav.
11. TOiP (технічне обслуговування) [Електронний ресурс] – Режим доступу до ресурсу: https://smart-eam.com/ua/news/toir-tehnicheskoe-obsluzhivanie-i-remonti/.

12. Міждержавні стандарти [Електронний ресурс] – Режим доступу до ресурсу: http://www.leonorm.lviv.ua/Default.php?Page =stlist&ObjId=76&CatId=6.

13. Пермякова О.О., Гаманюк І.М. Переваги використання СКБД SQLITE в застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, с.117-119

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОБЛІКУ ПЕРІОДИЧНОСТІ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ ТЕЛЕКОМУНІКАЦІЙНОГО ОБЛАДНАННЯ МОВОЮ C#

Виконала студентка 4 курсу
Групи ПД-43
Пермякова Ольга Олександрівна
Керівник роботи
Старший викладач кафедри ІПЗ Гаманюк Ігор Михайлович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення процесу обліку періодичності технічного обслуговування телекомунікаційного обладнання шляхом впровадження застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.
- **Об'єкт дослідження** – процес обліку періодичності технічного обслуговування телекомунікаційного обладнання.
- **Предмет дослідження** – застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз предметної галузі з обліку періодичності технічного обслуговування телекомунікаційного обладнання.
2. Провести аналіз інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання та визначити вимоги до розроблюваного застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.
3. Провести огляд засобів, які будуть використані при розробці застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.
4. Спроектувати застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.
5. Реалізувати застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.
6. Провести тестування розробленого застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

3

АНАЛІЗ ІСТРУМЕНТАЛЬНИХ ЗАСОБІВ

	Microsoft Excel	Windows Notepad	Hippo CMMS	Розроблений застосунок
Українська локалізація	+	+	-	+
Можливість доступу без підключення до інтернету	+	+	-	+
Автоматичне структурування даних	-	-	+	+
Фільтрування проведених технічних обслуговувань та телекомунікаційного обладнання, які відображаються користувачу	-	-	-	+
Пошук обладнання за інвентарним номером та моделлю	+	-	+	+
Пошук проведеного технічного обслуговування за датою проведення	+	-	+	+

4

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні вимоги:

1. внесення в систему даних про телекомунікаційне обладнання (тип, модель, інвентарний номер, MAC адреса, тощо);
2. додавання нових виконаних технічних обслуговувань;
3. можливість перегляду усього доданого телекомунікаційного обладнання;
4. можливість перегляду виконаних технічних обслуговувань;
5. можливість перегляду повної інформації про обрану одиницю обладнання;
6. засоби для роботи з типами технічних обслуговувань та даними користувачів (додавання, видалення, редагування);
7. пошук необхідної одиниці телекомунікаційного обладнання за інвентарним номером та моделлю;
8. пошук виконаного технічного обслуговування за датою проведення;
9. можливість фільтрування за типом телекомунікаційного обладнання та видом технічного обслуговування, при перегляді виконаних технічних обслуговувань.

Нефункціональні вимоги:

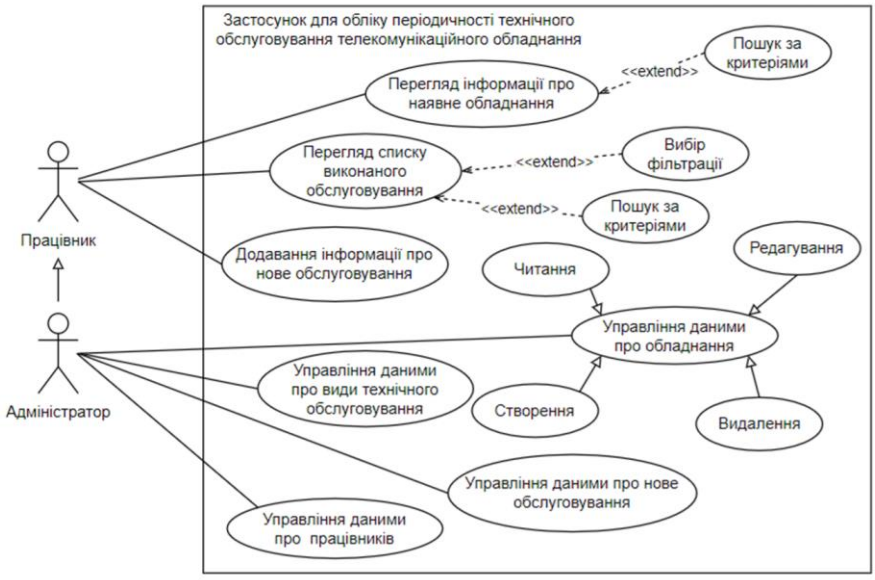
1. українська локалізація;
2. застосунок повинен відповідати на запити користувача в межах 2 секунд;
3. підтримка платформи Windows 10 і вище.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

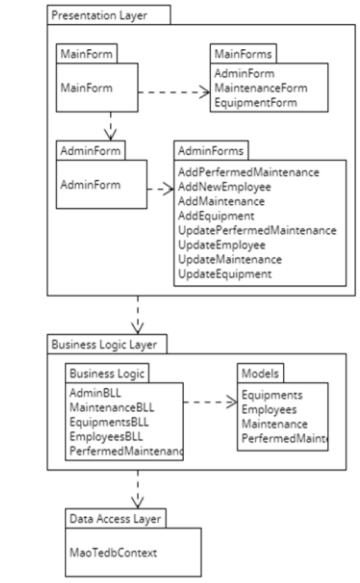
6

ДІАГРАМА ПРЕЦЕДЕНТІВ



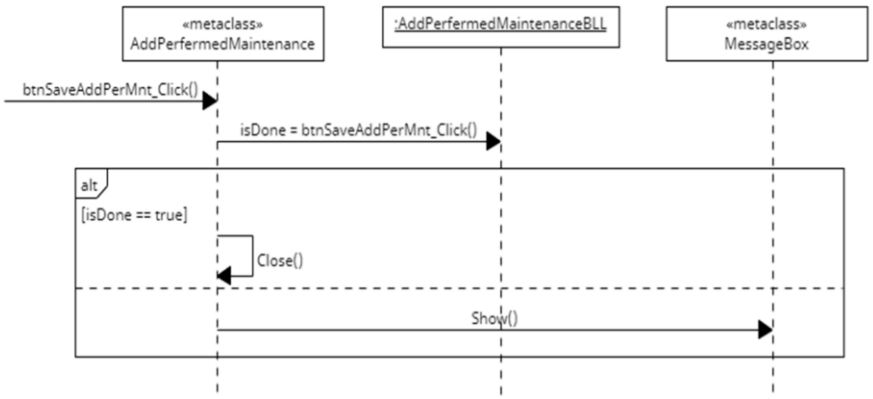
7

ДІАГРАМА ПАКЕТІВ



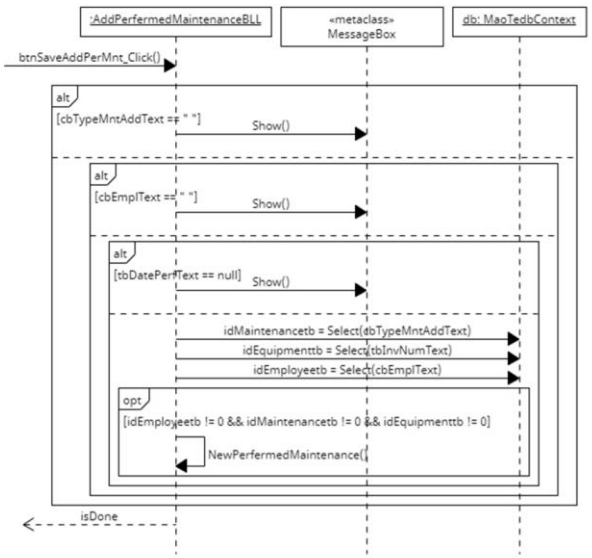
8

ДІАГРАМА ПОСЛІДОВНОСТІ



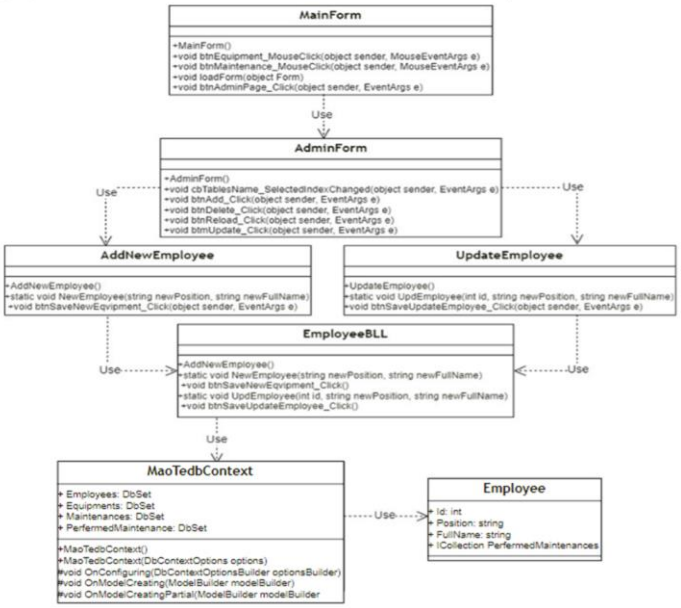
9

ДІАГРАМА ПОСЛІДОВНОСТІ



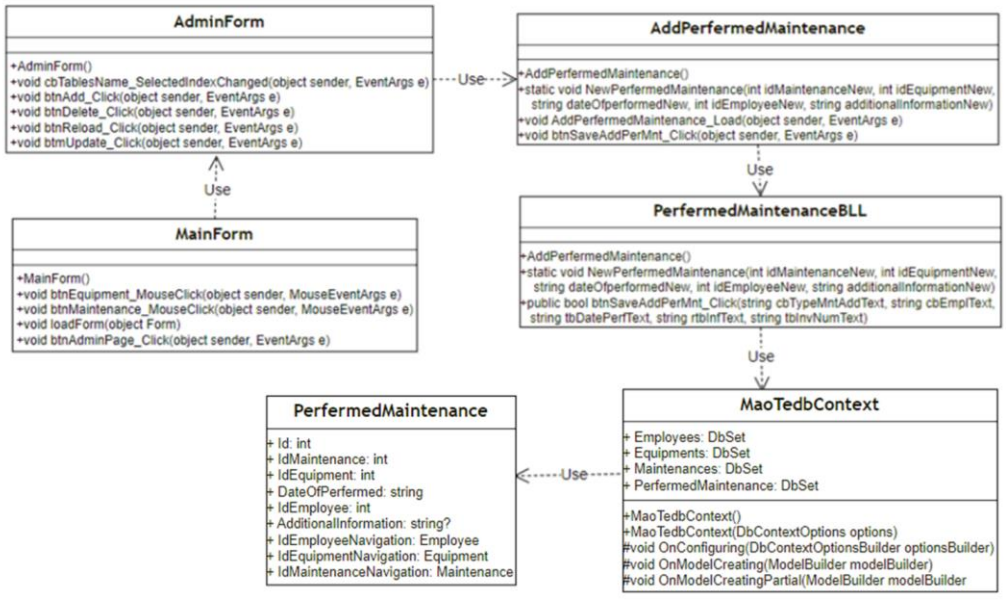
10

ДІАГРАМА КЛАСІВ ПРАЦІВНИКА



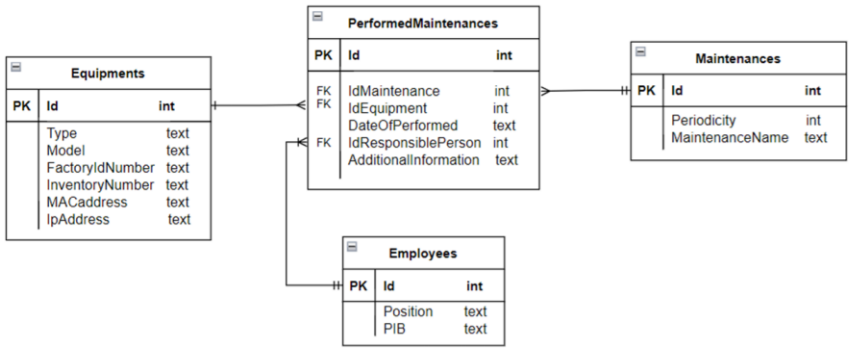
11

ДІАГРАМА КЛАСІВ ВИКОНАНОГО ОБСЛУГОВУВАННЯ



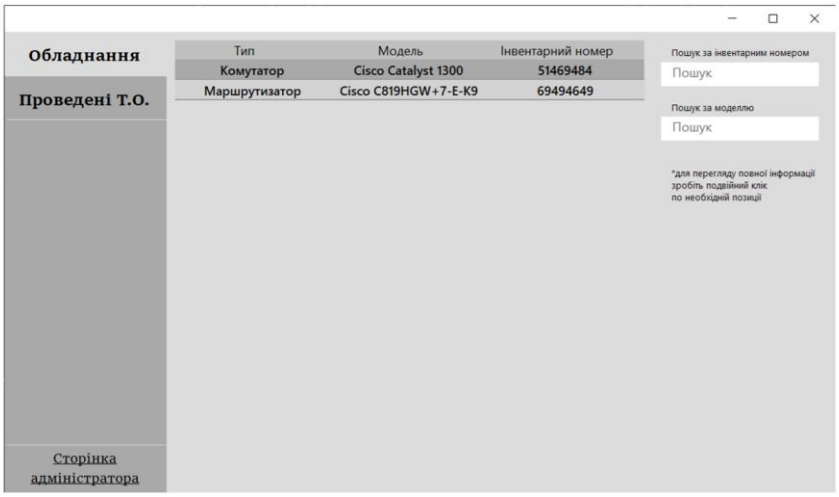
12

ER-ДІАГРАМА БАЗИ ДАНИХ



13

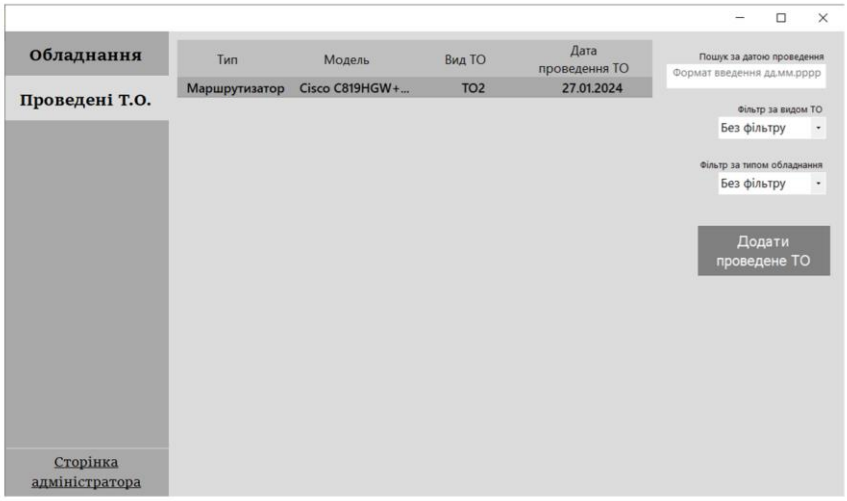
ЕКРАННІ ФОРМИ



Перегляд наявного телекомунікаційного обладнання

14

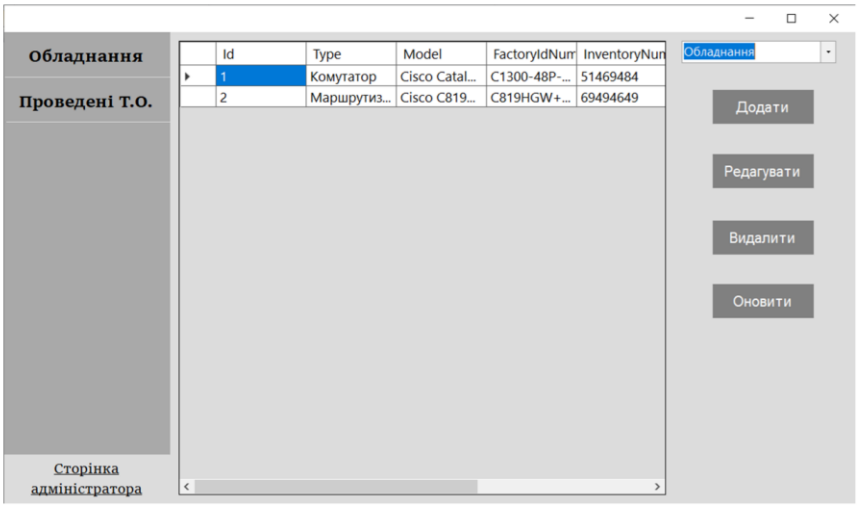
ЕКРАННІ ФОРМИ



Перегляд проведених технічних обслуговувань

15

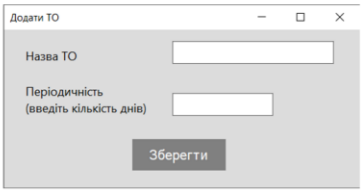
ЕКРАННІ ФОРМИ



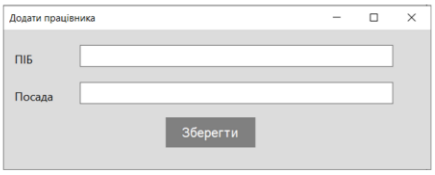
Сторінка адміністратора

16

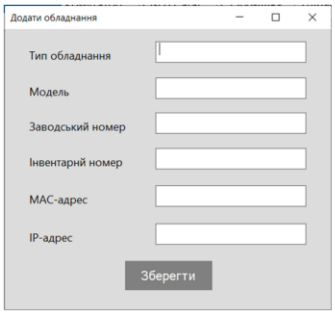
ЕКРАННІ ФОРМИ



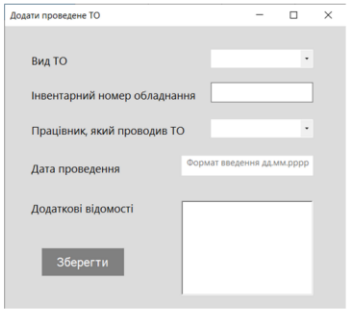
Додавання виду технічного обслуговування



Додавання працівника



Додавання обладнання



Додавання проведеного технічного обслуговування

17

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Пермякова О.О., Гаманюк І.М. Використання шаблону MVVM в застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, с.54-56

2. Пермякова О.О., Гаманюк І.М. Переваги використання СКБД SQLITE в застосунку для обліку періодичності технічного обслуговування телекомунікаційного обладнання. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, с.117-119

19

ВИСНОВКИ

1. Проведено аналіз предметної області, виявлено основні проблеми: велика кількість телекомунікаційного обладнання може ускладнити системним адміністраторам виявлення потенційних проблем; відсутність доступу до Інтернету унеможливлює системним адміністраторам забезпечити безперервну та безперебійну роботу мережевого та серверного обладнання.

2. Визначено функціональні та нефункціональні вимоги до розроблюваного застосунку. Основними функціональними вимогами є: редагування, видалення та додавання нового обладнання, працівників, виконаних технічних обслуговувань та видів технічних обслуговувань, а також пошук та фільтрація даних для відображення.

3. Проведено аналіз інструментальних засобів для обліку періодичності технічного обслуговування телекомунікаційного обладнання. За результатами аналізу визначено, що створення застосунку можна здійснювати мовою C#, за допомогою середовища розробки Visual Studio та платформи .NET Framework. Для створення інтерфейсу користувача було використано графічну підсистему WinForms. В якості системи керування базою даних було обрано SQLite та EntityFramework Core для взаємодії з нею.

4. Спроектовано та розроблено застосунок для обліку періодичності технічного обслуговування телекомунікаційного обладнання.

5. Проведено функціональне тестування з метою перевірки застосунку на відповідність визначеним вимогам.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
namespace MAoTE.Models;

public partial class Employee
{
    public int Id { get; set; }

    public string Position { get; set; } = null!;

    public string FullName { get; set; } = null!;

    public virtual ICollection<PerfermedMaintenance>
PerfermedMaintenances { get; set; } = new
List<PerfermedMaintenance>();
}
namespace MAoTE.Models;

public partial class Equipment
{
    public int Id { get; set; }

    public string Type { get; set; } = null!;

    public string Model { get; set; } = null!;

    public string FactoryIdNumber { get; set; } = null!;

    public string InventoryNumber { get; set; } = null!;

    public string? Macaddress { get; set; }

    public string? IpAddress { get; set; }

    public virtual ICollection<PerfermedMaintenance>
PerfermedMaintenances { get; set; } = new
List<PerfermedMaintenance>();
}
namespace MAoTE.Models;

public partial class Maintenance
{
    public int Id { get; set; }

    public int Periodicity { get; set; }

    public string MaintenanceName { get; set; } = null!;

    public virtual ICollection<PerfermedMaintenance>
PerfermedMaintenances { get; set; } = new
List<PerfermedMaintenance>();
}

namespace MAoTE.Models;

public partial class PerfermedMaintenance
{
    public int Id { get; set; }

    public int IdMaintenance { get; set; }

    public int IdEquipment { get; set; }

    public string DateOfPerfermed { get; set; } = null!;

    public int IdEmployee { get; set; }

    public string? AdditionalInformation { get; set; }

    public virtual Employee IdEmployeeNavigation { get;
set; } = null!;

    public virtual Equipment IdEquipmentNavigation {
get; set; } = null!;

    public virtual Maintenance IdMaintenanceNavigation
{ get; set; } = null!;
}

using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

namespace MAoTE.Models;

public partial class MaoTedbContext : DbContext
{
    public MaoTedbContext()
    {
    }
    public
MaoTedbContext(DbContextOptions<MaoTedbContext
> options)
        : base(options)
    {
    }

    public virtual DbSet<Employee> Employees { get;
set; }

    public virtual DbSet<Equipment> Equipments { get;
set; }

    public virtual DbSet<Maintenance> Maintenances {
get; set; }

    public virtual DbSet<PerfermedMaintenance>
PerfermedMaintenances { get; set; }

    protected override void
OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
```

```
    => optionsBuilder.UseSqlite("Data
Source=.\\DataBase\\MaoTEDB.db");

    protected override void
OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Employee>(entity =>
        {
            entity.HasIndex(e => e.Id,
"IX_Employees_Id").IsUnique();

            entity.Property(e =>
e.Position).HasColumnName("position");
            entity.Property(e =>
e.FullName).HasColumnName("FullName");


        });

        modelBuilder.Entity<Equipment>(entity =>
        {
            entity.HasIndex(e => e.Id,
"IX_Equipments_Id").IsUnique();

            entity.Property(e =>
e.FactoryIdNumber).HasColumnName("factoryIdNumbe
r");
            entity.Property(e =>
e.InventoryNumber).HasColumnName("inventoryNumb
er");
            entity.Property(e =>
e.IpAddress).HasColumnName("ipAddress");
            entity.Property(e =>
e.Macaddress).HasColumnName("MACAddress");
            entity.Property(e =>
e.Model).HasColumnName("model");
            entity.Property(e =>
e.Type).HasColumnName("type");
        });


        modelBuilder.Entity<Maintenance>(entity =>
        {
            entity.HasIndex(e => e.Id,
"IX_Maintenances_Id").IsUnique();

            entity.HasIndex(e => e.MaintenanceName,
"IX_Maintenances_maintenanceName").IsUnique();

            entity.Property(e =>
e.MaintenanceName).HasColumnName("maintenanceNa
me");
            entity.Property(e =>
e.Periodicity).HasColumnName("periodicity");
        });


modelBuilder.Entity<PerfermedMaintenance>(entity =>
        {
            entity.ToTable("PerfermedMaintenance");

            entity.HasIndex(e => e.Id,
"IX_PerfermedMaintenance_Id").IsUnique();

            entity.Property(e =>
e.AdditionalInformation).HasColumnName("additionalI
nformation");
            entity.Property(e =>
e.DateOfPerfermed).HasColumnName("dateOfPerferme
d");
            entity.Property(e =>
e.IdEquipment).HasColumnName("idEquipment");
            entity.Property(e =>
e.IdMaintenance).HasColumnName("idMaintenance");

            entity.HasOne(d =>
d.IdEmployeeNavigation).WithMany(p =>
p.PerfermedMaintenances)
                .HasForeignKey(d => d.IdEmployee)
                .OnDelete(DeleteBehavior.ClientSetNull);

            entity.HasOne(d =>
d.IdEquipmentNavigation).WithMany(p =>
p.PerfermedMaintenances)
                .HasForeignKey(d => d.IdEquipment)
                .OnDelete(DeleteBehavior.ClientSetNull);

            entity.HasOne(d =>
d.IdMaintenanceNavigation).WithMany(p =>
p.PerfermedMaintenances)
                .HasForeignKey(d => d.IdMaintenance)
                .OnDelete(DeleteBehavior.ClientSetNull);
        });

        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder
modelBuilder);
}

using MAoTE.Models;

namespace MAoTE.View
{
    public partial class AdminForm : Form
    {
        public AdminForm()
        {
            InitializeComponent();
        }


        private void
cbTablesName_SelectedIndexChanged(object sender,
EventArgs e)
        {
            MaoTedbContext db = new MaoTedbContext();

            switch (cbTablesName.Text)
            {
                case "Проведене ТО":
                    dgvMain.DataSource =
db.PerfermedMaintenances.ToList();
                    btnAdd.Visible = true;
                    break;
```

```
            case "Обладнання":
                dgvMain.DataSource =
db.Equipments.ToList();
                btnAdd.Visible = true;
                break;
            case "Працівники":
                dgvMain.DataSource =
db.Employees.ToList();
                btnAdd.Visible = true;
                break;
            case "TO":
                dgvMain.DataSource =
db.Maintenances.ToList();
                btnAdd.Visible = true;
                break;
            default:
                dgvMain.DataSource = null;
                btnAdd.Visible = true;
                break;
        }
    }

    private void btnAdd_Click(object sender,
EventArgs e)
    {

        switch (cbTablesName.Text)
        {
            case "Проведене ТО":
                Form f = new AddPerfermedMaintenance()
as Form;
                f.Show();
                MaoTedbContext db = new
MaoTedbContext();
                dgvMain.DataSource =
db.PerfermedMaintenances.ToList();
                break;
            case "Обладнання":
                Form f1 = new AddEquipment() as Form;
                f1.Show();
                MaoTedbContext db1 = new
MaoTedbContext();
                dgvMain.DataSource =
db1.Equipments.ToList();
                break;
            case "Працівники":
                Form f2 = new AddNewEmployee() as
Form;
                f2.Show();
                MaoTedbContext db2 = new
MaoTedbContext();
                dgvMain.DataSource =
db2.Employees.ToList();
                break;
            case "TO":
                Form f3 = new AddMaintenance() as Form;
                f3.Show();
                MaoTedbContext db3 = new
MaoTedbContext();
                dgvMain.DataSource =
db3.Maintenances.ToList();
                break;
            default:
                MessageBox.Show("Оберіть таблицю");
                break;
        }
    }

    private void btnDelete_Click(object sender,
EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();

        DataGridViewRow selectedRow =
dgvMain.CurrentRow;
        string selectedId =
selectedRow.Cells["Id"].Value.ToString();
        int idFromDgv = Int32.Parse(selectedId);
        DialogResult dialogResult =
MessageBox.Show("Видалити рядок з Id: " +
idFromDgv, "Повідомлення",
MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            switch (cbTablesName.Text)
            {
                case "Проведене ТО":
                    var selectedPerfMaintenance =
db.PerfermedMaintenances.FirstOrDefault(pm => pm.Id
== idFromDgv);

db.PerfermedMaintenances.Remove(selectedPerfMainte
nance);
                    db.SaveChanges();
                    dgvMain.DataSource =
db.PerfermedMaintenances.ToList();
                    break;
                case "Обладнання":
                    var selectedEquipment =
db.Equipments.FirstOrDefault(eq => eq.Id ==
idFromDgv);

db.Equipments.Remove(selectedEquipment);
                    db.SaveChanges();
                    dgvMain.DataSource =
db.Equipments.ToList();
                    break;
                case "Працівники":
                    var selectedEmployee =
db.Employees.FirstOrDefault(em => em.Id ==
idFromDgv);

db.Employees.Remove(selectedEmployee);
                    db.SaveChanges();
                    dgvMain.DataSource =
db.Employees.ToList();
                    break;
                case "TO":
                    var selectedMaintenance =
db.Maintenances.FirstOrDefault(mn => mn.Id ==
idFromDgv);

db.Maintenances.Remove(selectedMaintenance);
                    db.SaveChanges();
                    dgvMain.DataSource =
db.Maintenances.ToList();
```

```
                break;
            default:
                MessageBox.Show("Оберіть таблицю");
                break;
        }
    }
}

    private void btnReload_Click(object sender,
EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();

        switch (cbTablesName.Text)
        {
            case "Проведене ТО":
                dgvMain.DataSource =
db.PerfermedMaintenances.ToList();
                btnAdd.Visible = true;
                break;
            case "Обладнання":
                dgvMain.DataSource =
db.Equipments.ToList();
                btnAdd.Visible = true;
                break;
            case "Працівники":
                dgvMain.DataSource =
db.Employees.ToList();
                btnAdd.Visible = true;
                break;
            case "ТО":
                dgvMain.DataSource =
db.Maintenances.ToList();
                btnAdd.Visible = true;
                break;
            default:
                dgvMain.DataSource = null;
                btnAdd.Visible = true;
                break;

        }
    }

    private void btmUpdate_Click(object sender,
EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();

        DataGridViewRow selectedRow =
dgvMain.CurrentRow;
        string selectedId =
selectedRow.Cells["Id"].Value.ToString();
        int idFromDgv = Int32.Parse(selectedId);
        switch (cbTablesName.Text)
        {
            case "Проведене ТО":
                Form f = new
UpdatePerfermedMaintenance() as Form;
                f.Controls["tbId"].Text =
dgvMain.CurrentRow.Cells["Id"].Value.ToString();
                f.Controls["tbMaintType"].Text =
dgvMain.CurrentRow.Cells["IdMaintenance"].Value.To
String();
                f.Controls["tbInvNum"].Text =
dgvMain.CurrentRow.Cells["IdEquipment"].Value.ToStr
ing();
                f.Controls["tbEmployee"].Text =
dgvMain.CurrentRow.Cells["IdEmployee"].Value.ToStri
ng();
                f.Controls["tbDatePerf"].Text =
dgvMain.CurrentRow.Cells["DateOfPerfermed"].Value.
ToString();
                f.Controls["rtbInf"].Text =
dgvMain.CurrentRow.Cells["AdditionalInformation"].V
alue.ToString();
                f.Show();
                MaoTedbContext db0 = new
MaoTedbContext();
                dgvMain.DataSource =
db0.PerfermedMaintenances.ToList();
                break;
            case "Обладнання":
                Form f1 = new UpdateEquipment() as Form;
                f1.Controls["tbId"].Text =
dgvMain.CurrentRow.Cells["Id"].Value.ToString();
                f1.Controls["tbTypeNewEqp"].Text =
dgvMain.CurrentRow.Cells["Type"].Value.ToString();
                f1.Controls["tbModel"].Text =
dgvMain.CurrentRow.Cells["Model"].Value.ToString();
                f1.Controls["tbFactNum"].Text =
dgvMain.CurrentRow.Cells["FactoryIdNumber"].Value.
ToString();
                f1.Controls["tbInvNumNewEqp"].Text =
dgvMain.CurrentRow.Cells["InventoryNumber"].Value.
ToString();
                f1.Controls["tbMACaddress"].Text =
dgvMain.CurrentRow.Cells["Macaddress"].Value.ToStri
ng();
                f1.Controls["tbIPaddress"].Text =
dgvMain.CurrentRow.Cells["IpAddress"].Value.ToStrin
g();
                f1.Show();
                MaoTedbContext db1 = new
MaoTedbContext();
                dgvMain.DataSource =
db1.Equipments.ToList();
                break;
            case "Працівники":
                Form f2 = new UpdateEmployee() as Form;
                f2.Controls["tbId"].Text =
dgvMain.CurrentRow.Cells["Id"].Value.ToString();
                f2.Controls["tbFullName"].Text =
dgvMain.CurrentRow.Cells["FullName"].Value.ToStrin
g();
                f2.Controls["tbPosition"].Text =
dgvMain.CurrentRow.Cells["Position"].Value.ToString()
;
                f2.Show();
                MaoTedbContext db2 = new
MaoTedbContext();
                dgvMain.DataSource =
db2.Employees.ToList();
                break;
            case "ТО":
                Form f3 = new UpdateMaintenance() as
Form;
```

```
                f3.Controls["tbId"].Text =
dgvMain.CurrentRow.Cells["Id"].Value.ToString();
                f3.Controls["tbNameMaint"].Text =
dgvMain.CurrentRow.Cells["MaintenanceName"].Value
.ToString();
                f3.Controls["tbPeriodicity"].Text =
dgvMain.CurrentRow.Cells["Periodicity"].Value.ToStrin
g();
                f3.Show();
                MaoTedbContext db3 = new
MaoTedbContext();
                dgvMain.DataSource =
db3.Maintenances.ToList();
                break;
            default:
                MessageBox.Show("Оберіть таблицю");
                break;
        }
    }
  }
}

using MAoTE.Models;

namespace MAoTE.View
{
  public partial class AddEquipment : Form
  {
    public AddEquipment()
    {
      InitializeComponent();
    }

    public static void NewEquipment(string newType,
string newModel, string newFactoryNum, string
newInvNum, string newMac, string newIp)
    {
      using (var db = new MaoTedbContext())
      {
        var std = new Equipment { Type = newType,
Model = newModel, FactoryIdNumber =
newFactoryNum, InventoryNumber = newInvNum,
Macaddress = newMac, IpAddress = newIp};
        db.Equipments.Add(std);
        db.SaveChanges();
      }
    }

    private void btnSaveNewEqvipment_Click(object
sender, EventArgs e)
    {
      if (tbTypeNewEqp.Text == "")
      {
        MessageBox.Show("Введіть тип
обладнання!");
      }
      else
      {
        if (tbModel.Text == "")
        {
          MessageBox.Show("Введіть модель!");
        }
        else
        {
          if (tbFactNum.Text == "")
          {
            MessageBox.Show("Введіть заводський
номер!");
          }
          else
          {
            if (tbInvNumNewEqp.Text == "")
            {
              MessageBox.Show("Введіть
інвентарний номер!");
            }
            else
            {
              NewEquipment(tbTypeNewEqp.Text,
tbModel.Text, tbFactNum.Text, tbInvNumNewEqp.Text,
tbMACaddress.Text, tbIPaddress.Text);
              this.Close();
            }
          }
        }
      }
    }
  }
}

using MAoTE.Models;

namespace MAoTE.View
{
  public partial class AddMaintenance : Form
  {
    public AddMaintenance()
    {
      InitializeComponent();
    }

    public static void NewMaintenance(int
newPeriodicity, string newMaintenanceName)
    {
      using (var db = new MaoTedbContext())
      {
        var std = new Maintenance { Periodicity =
newPeriodicity, MaintenanceName =
newMaintenanceName };
        db.Maintenances.Add(std);
        db.SaveChanges();
      }
    }
    private void btnSaveNewEqvipment_Click(object
sender, EventArgs e)
    {
      if (tbNameMaint.Text == "")
      {
        MessageBox.Show("Введіть назву ТО!");
      }
      else
      {
        if(tbPeriodicity.Text == "")
        {
          MessageBox.Show("Введіть періодичність
проведення ТО!");
```

```
            }
            else
            {
                int periodicity =
Convert.ToInt32(tbPeriodicity.Text);
                NewMaintenance(periodicity,
tbNameMaint.Text);
            }
        }
    }
  }
}

using MAoTE.Models;

namespace MAoTE.View
{
  public partial class AddNewEmployee : Form
  {
    public AddNewEmployee()
    {
      InitializeComponent();
    }

    public static void NewEmployee(string
newPosition, string newFullName)
    {
      using (var db = new MaoTedbContext())
      {
        var std = new Employee { Position =
newPosition, FullName = newFullName };
        db.Employees.Add(std);
        db.SaveChanges();
      }
    }

    private void btnSaveNewEqvipment_Click(object
sender, EventArgs e)
    {
      MaoTedbContext db = new MaoTedbContext();

      if (tbFullName.Text == "")
      {
        MessageBox.Show("Введіть прізвище, ім'я,
по батькові!");
      }
      else
      {
        if (tbPosition.Text == "")
        {
          MessageBox.Show("Введіть посаду!");
        }
        else
        {
          NewEmployee(tbPosition.Text,
tbFullName.Text);
          this.Close();
        }
      }
    }
  }
}
```

```
using MAoTE.Models;
using System.Data;


namespace MAoTE.View
{
  public partial class AddPerfermedMaintenance : Form
  {
    public AddPerfermedMaintenance()
    {
      InitializeComponent();
    }

    public static void NewPerfermedMaintenance(int
idMaintenanceNew, int idEquipmentNew, string
dateOfperformedNew, int idEmployeeNew, string
additionalInformationNew)
    {

      using (var db = new MaoTedbContext())
      {
        var std = new PerfermedMaintenance {
IdMaintenance = idMaintenanceNew, IdEquipment =
idEquipmentNew, DateOfPerfermed =
dateOfperformedNew, IdEmployee = idEmployeeNew,
AdditionalInformation = additionalInformationNew };
        db.PerfermedMaintenances.Add(std);
        db.SaveChanges();
      }

    }

    private void
AddPerfermedMaintenance_Load(object sender,
EventArgs e)
    {
      MaoTedbContext db = new MaoTedbContext();

      var MnTypes = db.Maintenances.Select(mn =>
mn.MaintenanceName).ToList();
      MnTypes.Insert(0, " ");
      cbTypeMntAdd.DataSource = MnTypes;

      var EmplFullName = db.Employees.Select(mn
=> mn.FullName).ToList();
      EmplFullName.Insert(0, " ");
      cbEmpl.DataSource = EmplFullName;
    }

    private void btnSaveAddPerMnt_Click(object
sender, EventArgs e)
    {
      MaoTedbContext db = new MaoTedbContext();

      if(cbTypeMntAdd.Text == " ")
      {
        MessageBox.Show("Оберіть тип ТО із
списку.");
      }
      else
      {
        if(cbEmpl.Text == " ")
        {
```

```
                MessageBox.Show("Оберіть працівника,
який здійснив ТО, із списку.");
            }
            else
            {
                if(tbDatePerf.Text == null)
                {
                    MessageBox.Show("Введіть дату коли
було виконано ТО в форматі дд.мм.рррр");
                }
                else
                {
                    int idMaintenancetb = (from ma in
db.Maintenances
                                           where
ma.MaintenanceName == cbTypeMntAdd.Text
                                           select ma.Id).First();

                    int idEquipmenttb = (from eq in
db.Equipments
                                         where eq.InventoryNumber
== tbInvNum.Text
                                         select eq.Id).First();

                    int idEmployeetb = (from em in
db.Employees
                                        where em.FullName ==
cbEmpl.Text
                                        select em.Id).First();

                    if (idMaintenancetb != 0)
                    {
                        if(idEquipmenttb != 0)
                        {
                            if(idEmployeetb != 0)
                            {

NewPerfermedMaintenance(idMaintenancetb,
idEquipmenttb, tbDatePerf.Text, idEmployeetb,
rtbInf.Text);
                                this.Close();
                            }
                            else
                            {
                                MessageBox.Show("Введіть
коректні дані!");
                            }
                        }
                    }
                }
            }
        }
    }
}

using MAoTE.Models;
using System.Data;

namespace MAoTE.View
{
    public partial class EquipmentForm : Form
    {
        public EquipmentForm()
        {
            InitializeComponent();
        }

        private void EquipmentForm_Load(object sender,
EventArgs e)
        {
            MaoTedbContext db = new MaoTedbContext();

            dgvEquipment.DataSource = (from eq in
db.Equipments
                                       select new { eq.Type, eq.Model,
eq.InventoryNumber }).ToList();
        }

        private void tbInvSearch_KeyDown(object sender,
KeyEventArgs e)
        {
            MaoTedbContext db = new MaoTedbContext();
            if (e.KeyCode == Keys.Enter)
            {
                e.SuppressKeyPress = true;

                dgvEquipment.DataSource = (from eq in
db.Equipments
                                           where eq.InventoryNumber
== tbInvSearch.Text
                                           select new { eq.Type,
eq.Model, eq.InventoryNumber }).ToList();
            }
            else
            {
                dgvEquipment.DataSource = (from eq in
db.Equipments
                                           select new { eq.Type,
eq.Model, eq.InventoryNumber }).ToList();
            }
        }

        private void tbModelSearch_KeyDown(object
sender, KeyEventArgs e)
        {
            MaoTedbContext db = new MaoTedbContext();
            if (e.KeyCode == Keys.Enter)
            {
                e.SuppressKeyPress = true;

                dgvEquipment.DataSource = (from eq in
db.Equipments
                                           where eq.Model ==
tbModelSearch.Text
                                           select new { eq.Type,
eq.Model, eq.InventoryNumber }).ToList();
            }
            else
            {
                dgvEquipment.DataSource = (from eq in
db.Equipments
                                           select new { eq.Type,
eq.Model, eq.InventoryNumber }).ToList();
            }
        }
```

```
        private void dgvEquipment_DoubleClick(object
sender, EventArgs e)
        {
            MaoTedbContext db = new MaoTedbContext();
            DataGridViewRow selectedRow =
dgvEquipment.CurrentRow;
            string selectedInventoryNumber =
selectedRow.Cells["InventoryNumber"].Value.ToString(
);
            var selectedEquipment =
db.Equipments.FirstOrDefault(eq =>
eq.InventoryNumber == selectedInventoryNumber);

            if (selectedEquipment != null)
            {
                string message = $"Інвентарний номер:
{selectedEquipment.InventoryNumber}\n" +
                                 $"Тип обладнання:
{selectedEquipment.Type}\n" +
                                 $"Модель:
{selectedEquipment.Model}\n" +
                                 $"Заводський номер:
{selectedEquipment.FactoryIdNumber}\n" +
                                 $"MAC-адрес:
{selectedEquipment.Macaddress ?? "N/A"}\n" +
                                 $"IP-адрес:
{selectedEquipment.IpAddress ?? "N/A"}";

                MessageBox.Show(message, "Повна
інформація про обладнання");
            }
        }
    }
}

using MAoTE.View;

namespace MAoTE
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
            loadForm(new EquipmentForm());
            btnEquipment.BackColor = Color.Gainsboro;
        }

        private void btnEquipment_MouseClick(object
sender, MouseEventArgs e)
        {
            btnEquipment.BackColor = Color.Gainsboro;
            btnMaintenance.BackColor = Color.DarkGray;
            btnAdminPage.BackColor = Color.DarkGray;

            loadForm(new EquipmentForm());
        }

        private void btnMaintenance_MouseClick(object
sender, MouseEventArgs e)
        {
            btnMaintenance.BackColor = Color.Gainsboro;
            btnEquipment.BackColor = Color.DarkGray;
            btnAdminPage.BackColor = Color.DarkGray;

            loadForm(new MaintenanceForm());
        }

        public void loadForm(object Form)
        {
            if (this.MainPanel.Controls.Count > 0)
            {
                this.MainPanel.Controls.RemoveAt(0);
            }
            Form f = Form as Form;
            f.TopLevel = false;
            f.Dock = DockStyle.Fill;
            this.MainPanel.Controls.Add(f);
            this.MainPanel.Tag = f;
            f.Show();
        }

        private void btnAdminPage_Click(object sender,
EventArgs e)
        {
            btnAdminPage.BackColor = Color.Gainsboro;
            btnEquipment.BackColor = Color.DarkGray;
            btnMaintenance.BackColor = Color.DarkGray;

            loadForm(new AdminForm());
        }
    }
}

using MAoTE.Models;
using System.Data;
namespace MAoTE.View
{
    public partial class MaintenanceForm : Form
    {

        public MaintenanceForm()
        {
            InitializeComponent();
            MaoTedbContext db = new MaoTedbContext();
        }
        private void AllPerfermedfMaintenance()
        {
            MaoTedbContext db = new MaoTedbContext();


            dgvPerfMain.DataSource = (from pm in
db.PerfermedMaintenances
                                      join m in db.Maintenances on
pm.IdMaintenance equals m.Id
                                      join eq in db.Equipments on
pm.IdEquipment equals eq.Id
                                      select new { eq.Type, eq.Model,
m.MaintenanceName, pm.DateOfPerfermed }).ToList();
        }
        private void MaintenanceForm_Load(object sender,
EventArgs e)
        {

            AllPerfermedfMaintenance();
```

```
        MaoTedbContext db = new MaoTedbContext();

        var MnTypes = db.Maintenances.Select(mn =>
mn.MaintenanceName).ToList();
        MnTypes.Insert(0, "Без фільтру");
        cbTypeMn.DataSource = MnTypes;

        var EqTypes = db.Equipments.Select(eq =>
eq.Type).Distinct().ToList();
        EqTypes.Insert(0, "Без фільтру");
        cbTypeEq.DataSource = EqTypes;
    }

    private void cbType_SelectedIndexChanged(object
sender, EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();

        if (cbTypeMn.Text != "Без фільтру")
        {
            dgvPerfMain.DataSource = (from pm in
db.PerfermedMaintenances
                                      join m in db.Maintenances on
pm.IdMaintenance equals m.Id
                                      join eq in db.Equipments on
pm.IdEquipment equals eq.Id
                                      where m.MaintenanceName
== cbTypeMn.Text
                                      select new { eq.Type,
eq.Model, m.MaintenanceName, pm.DateOfPerfermed }
                                  ).ToList();
        }
        else
        {
            AllPerfermedfMaintenance();
        }
    }

    private void tbInvSearchDate_KeyDown(object
sender, KeyEventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();
        if (e.KeyCode == Keys.Enter)
        {
            e.SuppressKeyPress = true;

            dgvPerfMain.DataSource = (from pm in
db.PerfermedMaintenances
                                      join m in db.Maintenances on
pm.IdMaintenance equals m.Id
                                      join eq in db.Equipments on
pm.IdEquipment equals eq.Id
                                      where pm.DateOfPerfermed
== tbInvSearchDate.Text
                                      select new { eq.Type,
eq.Model, m.MaintenanceName, pm.DateOfPerfermed }
                                  ).ToList();
        }
        else
        {
            dgvPerfMain.DataSource = (from pm in
db.PerfermedMaintenances
                                      join m in db.Maintenances on
pm.IdMaintenance equals m.Id
                                      join eq in db.Equipments on
pm.IdEquipment equals eq.Id
                                      select new { eq.Type,
eq.Model, m.MaintenanceName, pm.DateOfPerfermed
}).ToList();
        }
    }
    private void
cbTypeEq_SelectedIndexChanged(object sender,
EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();

        if (cbTypeEq.Text != "Без фільтру")
        {
            dgvPerfMain.DataSource = (from pm in
db.PerfermedMaintenances
                                      join m in db.Maintenances on
pm.IdMaintenance equals m.Id
                                      join eq in db.Equipments on
pm.IdEquipment equals eq.Id
                                      where eq.Type ==
cbTypeEq.Text
                                      select new { eq.Type,
eq.Model, m.MaintenanceName, pm.DateOfPerfermed }
                                  ).ToList();
        }
        else
        {
            AllPerfermedfMaintenance();
        }
    }
    private void btnSaveAddPerMnt_Click(object
sender, EventArgs e)
    {
        Form f = new AddPerfermedMaintenance() as
Form;
        f.Show();
    }
  }
}

using MAoTE.Models;
namespace MAoTE.View
{
  public partial class UpdateEmployee : Form
  {
    public UpdateEmployee()
    {
      InitializeComponent();
    }

    public static void UpdEmployee(int id, string
newPosition, string newFullName)
    {
      using (var db = new MaoTedbContext())
      {
        var std = db.Employees.FirstOrDefault(e =>
e.Id == id);
        std.Position = newPosition;
        std.FullName = newFullName;
```

```
            db.SaveChanges();
        }
    }
    private void btnSaveUpdateEmployee_Click(object
sender, EventArgs e)
    {
        MaoTedbContext db = new MaoTedbContext();
        int id = Int32.Parse(tbId.Text);
        UpdEmployee(id, tbPosition.Text,
tbFullName.Text);
        this.Close();
    }
  }
}
using MAoTE.Models;
namespace MAoTE.View
{
  public partial class UpdateEquipment : Form
  {
    public UpdateEquipment()
    {
      InitializeComponent();
    }

    public static void UpdEquipment(int id, string
newType, string newModel, string newFactoryNum,
string newInvNum, string newMac, string newIp)
    {
      using (var db = new MaoTedbContext())
      {
        var std = db.Equipments.FirstOrDefault(e =>
e.Id == id);
        std.Type = newType;
        std.Model = newModel;
        std.Macaddress = newMac;
        std.IpAddress = newIp;
        std.FactoryIdNumber = newFactoryNum;
        std.InventoryNumber = newInvNum;
        db.SaveChanges();
      }
    }
    private void btnSaveUpdateEqvipment_Click(object
sender, EventArgs e)
    {
      int id = Int32.Parse(tbId.Text);
       UpdEquipment( id, tbTypeNewEqp.Text,
tbModel.Text, tbFactNum.Text, tbInvNumNewEqp.Text,
tbMACaddress.Text, tbIPaddress.Text);
       this.Close();
    }
  }
}

using MAoTE.Models;
namespace MAoTE.View
{
  public partial class UpdateMaintenance : Form
  {
    public UpdateMaintenance()
    {
      InitializeComponent();
    }

    public static void UpdMaintenance(int id, int
newPeriodicity, string newMaintenanceName)
    {
      using (var db = new MaoTedbContext())
      {
        var std = db.Maintenances.FirstOrDefault(e =>
e.Id == id);
        std.MaintenanceName =
newMaintenanceName;
        std.Periodicity = newPeriodicity;
        db.SaveChanges();
      }
    }
    private void
btnSaveUpdateMaintenance_Click(object sender,
EventArgs e)
    {
      int id = Int32.Parse(tbId.Text);
      int periodicity =
Convert.ToInt32(tbPeriodicity.Text);
      UpdMaintenance(id, periodicity,
tbNameMaint.Text);
    }
  }
}

using MAoTE.Models;

namespace MAoTE.View
{
  public partial class UpdatePerfermedMaintenance :
Form
  {
    public UpdatePerfermedMaintenance()
    {
      InitializeComponent();
    }
    public static void UpdPerfermedMaintenance(int id,
int idMaintenanceNew, int idEquipmentNew, string
dateOfperformedNew, int idEmployeeNew, string
additionalInformationNew)
    {
      using (var db = new MaoTedbContext())
      {
        var std =
db.PerfermedMaintenances.FirstOrDefault(e => e.Id ==
id);
        std.DateOfPerfermed = dateOfperformedNew;
        std.IdMaintenance = idMaintenanceNew;
        std.IdEquipment = idEquipmentNew;
        std.AdditionalInformation =
additionalInformationNew;
        std.IdEmployee = idEmployeeNew;
        db.SaveChanges();
      }
    }
    private void btnSaveAddPerMnt_Click(object
sender, EventArgs e)
    {
      MaoTedbContext db = new MaoTedbContext();

      int id = Int32.Parse(tbId.Text);
```

```
            int maintType =
Convert.ToInt32(tbMaintType.Text);
            int invNum = Convert.ToInt32(tbInvNum.Text);
            int employee =
Convert.ToInt32(tbEmployee.Text);

            UpdPerfermedMaintenance(id, maintType,
invNum, tbDatePerf.Text, employee, rtbInf.Text);
        }
    }
}
```