Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Центр _____Післядипломної освіти_____
(повна назва)

Кафедра _____програмної інженерії_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти _____перший (бакалаврський)_____

_____Програмна система обліку приладів залізничної автоматики____
(тема)

Виконав:
студент _4_ курсу, групи ___ПЗПІп-22-1____
Молявкін А.В._____
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного_
_____Забезпечення_____
(код і повна назва спеціальності)

Тип програми ___освітньо-професійна_____

Освітня програма _Програмна інженерія___
(повна назва освітньої програми)

Керівник _доц. Русакова Н.Є._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ___З.В. Дудар_____
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Центр _Післядипломної освіти_____
Кафедра _Програмної Інженерії_____
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність_121 – Інженерія програмного забезпечення_____
(код і повна назва)
Тип програми _____освітньо-професійна_____

Освітня програма _Програмна інженерія_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
___ _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Молявкіну Артему Валерійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) _Програмна система обліку приладів залізничної автоматики_____

затверджена наказом університету від 17 червня 2024 р. № 588Ст_____

2. Термін здачі студентом закінченої роботи 18___07___2024 р.

3. Вихідні дані до роботи *В програмній системі передбачити: підтримку виконання побудови інтерфейсу, описаної інструментами, наданими мовою програмування C# .*

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Позначка про виконання
1	Аналіз предметної галузі	20.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	20.06.2024	*виконано*
5	Тестування ПЗ	30.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.07.2024	*виконано*
7	Підготовка презентації та доповіді	06.07.2024	*виконано*
8	Попередній захист	18.07.2024	*виконано*
9	Нормоконтроль, рецензування	16.07.2024	*виконано*
10	Здача роботи у електронний архів	18.07.2024	*виконано*
11	Допуск до захисту у зав. Кафедри	18.07.2024	*виконано*

Дата видачі завдання _6_ _травня_____2024 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Русакова Н.Є.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка з кваліфікаційної роботи містить в собі: 70 сторінок, 41 рисунок, 1 таблицю.

ПРИЛАДИ ЗАЛІЗНИЧНОЇ АВТОМАТИКИ, СИСТЕМА ОБЛІКУ, C#, MS SQL, ADO .NET.

Об’єкт розробки – програмна система обліку приладів залізничної автоматики.

Метою кваліфікаційної роботи є застосування практичних навичок здобутих в період навчання для виконання кваліфікаційної роботи.

Мета розробки - підвищити рівень автоматизації процесів та зменшити кількість ручної праці, за рахунок централізованому збору та аналізу даних.

Метод рішення – середовище розробки Microsoft Visual Studio 2022, мова програмування C#, фреймворк ADO.NET, бібліотека для створення штрих-кодів BarcodeLib, СУБД MS SQL.

У результаті розробки створено програмну систему, що дозволяє автоматизувати процес обліку приладів залізничної автоматики та надавати відповідні звіти користувачам системи.

DEVICES OF RAILWAY AUTOMATION, ACCOUNTING SYSTEM, C#, MS SQL, ADO .NET.

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

The purpose of pre-certification practice is to apply the practical skills acquired during the training period to perform qualification work.

The purpose of the development is to increase the level of automation of processes and reduce the amount of manual work, due to the centralized collection and analysis of data.

Solution method – Microsoft Visual Studio 2022 development environment, C# programming language, ADO.NET framework, BarcodeLib library for creating barcodes, MS SQL DBMS.

As a result of the development, a software system was created that allows you to automate the process of accounting for railway automation devices and provide relevant reports to system users.

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

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

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

РТД – ремонтно-технологічна ділянка

КВП – контрольно-вимірювальний пункт

ШЧ – дистанція сигналізації та зв’язку

Ш – служба автоматики та телемеханіки

СЦБ – Сигналізація Централізація Блокування

ПЗ – програмне забезпечення

ІС – інформаційна система

ПС – програмна система

БД – база даних

СУБД – система управління базами даних

UML — Unified Modeling Language

ВСТУП

Залізничний транспорт відіграє ключову роль у забезпеченні стабільного функціонування економіки та ефективного переміщення пасажирів і вантажів. Безпечність та ефективність залізничних перевезень значною мірою залежить від стану та надійності залізничної автоматики, яка контролює і регулює рух потягів, забезпечує чітке функціонування сигнальних систем та інших критично важливих елементів інфраструктури.

Система обліку приладів залізничної автоматики є невід'ємною частиною сучасної залізничної інфраструктури. Вона забезпечує детальний моніторинг, контроль та управління всіма аспектами роботи приладів автоматики. Така система дозволяє в режимі реального часу відстежувати стан обладнання, планувати та виконувати профілактичні ремонти, запобігати аваріям та збоїв, а також оптимізувати витрати на обслуговування і модернізацію залізничної автоматики.

Розробка системи обліку приладів залізничної автоматики має на меті забезпечити ефективне управління, моніторинг та підтримку автоматизованих систем, які використовуються для забезпечення безпеки та ефективності залізничних перевезень.

Інтеграція з іншими системами управління залізничним транспортом та впровадження сучасних технологій дозволяють підвищити рівень автоматизації процесів та зменшити кількість ручної праці.

Метою роботи є створення програмної системи яка забезпечить надійну роботу приладів залізничної автоматики за рахунок постійного моніторингу стану приладів для своєчасного виявлення і усунення несправностей.

Впровадження системи обліку приладів залізничної автоматики дозволяє значно підвищити безпеку та надійність залізничного транспорту, зменшити експлуатаційні витрати та забезпечити високу ефективність роботи залізничної інфраструктури. Це є критично важливим кроком на шляху до побудови сучасної та інтелектуальної транспортної системи, яка відповідає вимогам XXI століття.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Комплекс завдань дорожнього рівня «Облік приладів та планування роботи ділянок РТД» призначається для старших механіків РТД, керівництва ШЧ та Ш, лінійних механіків, бригади комплексної заміни. Основним призначенням системи є планування, оптимізація та фіксація виконання ходу робіт із заміни та ремонту пристроїв СЦБ.

Система обліку приладів входить до складу комплексної автоматизованої системи управління господарством сигналізації, централізації та блокування.

Створення системи обліку ставить за мету підвищення якості та оперативності виконання робіт із заміни та ремонту пристроїв СЦБ, обґрунтованості прийняття рішень фахівцями та керівниками ШЧ та Ш шляхом автоматизації процесів планування, оптимізації та контролю виконання робіт.

Система обліку приладів забезпечує автоматизацію наступних функцій працівників господарства:

- створення та ведення інформації про конкретні прилади та про місце їх встановлення у складі єдиної бази даних колективного користування;
- супровід переміщень приладів з видачею технологічно необхідної інформації;
- контроль виконання планів заміни приладів;
- аналіз відмов приладів, що відбулися з вини РТД;
- планування індивідуальних завдань працівникам ділянок РТД;
- видача вихідних документів, що визначаються технологією роботи дорожнього та дистанційного рівнів;
- пошук приладів у БД за довільним запитом;
- оптимізація планування робіт із заміни приладів;
- оптимізація планування робіт з ремонту приладів;
- подання необхідної інформації для реалізації функцій управління інших систем обліку приладів;

- автоматичний обмін даними між системами обліку дорожнього та дистанційного рівнів та пересилання інформації до інших підрозділів та організацій.

Для спрощення роботи оператора КВП під час роботи з інформацією він повинен мати можливість:

- сортувати прилади по датам перевірки;
- здійснювати пошук інформації про прилади які встановлені на об’єкті;
- здійснювати фільтрацію інформації про прилади, які вимагають повторної перевірки в конкретному місяці;
- здійснювати пошук приладів які встановлені на об’єктах відповідної станції.

Описання алгоритмічних залежностей показників в ПЗ роботи системи обліку приладів: дата наступної перевірки приладу <= дата перевірки приладу + періодичність перевірки приладу.

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно ідентифікації:

- кожний прилад буде ідентифікуватися сурогатним ідентификатором, оскільки в обліку можуть знаходитися багато приладів одного типу;
- кожний об’єкт буде ідентифікуватися сурогатним ідентификатором, оскільки в межах однієї ділянки на різних станціях можуть знаходитися об’єкти з однаковою назвою.

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

- кожний працівник може перевірити декілька приладів;
- на одній станції можуть знаходитися декілька об’єктів;
- на одному об’єкті можуть знаходитися декілька приладів;
- у приладу в процесі експлуатації може бути декілька відмов.

1.2 Виявлення та вирішення проблем

Для аналізу було відібрано 5 систем-конкурентів: Infor EAM [1], IBM Maximo [2], UpKeep [3], Fiix [4], eMaint CMMS [5].

Infor EAM (Enterprise Asset Management) — це потужна система управління активами, яка забезпечує повний набір функцій для управління фізичними активами організації.

Основні функції Infor EAM включають:

- інвентаризація активів: облік та відстеження всіх активів організації, включаючи місцезнаходження, стан, вартість та інші атрибути;
- планове технічне обслуговування: планування та виконання регулярних обслуговувань для підтримки активів у належному стані;
- непланове технічне обслуговування: управління аварійними ремонтами та обслуговуванням;
- робочі замовлення: створення, призначення та відстеження робочих замовлень на обслуговування та ремонт активів;
- інвентаризація запасів: облік та управління запасами, необхідними для технічного обслуговування та ремонту;
- закупівлі та постачання: управління процесами закупівель та постачання запасів і матеріалів;
- контроль рівня запасів: відстеження та контроль рівня запасів для уникнення дефіциту або надлишків;
- звіти та дашборди: генерація звітів та дашбордів для аналізу ефективності управління активами, технічного обслуговування та витрат;
- інтеграція з іншими системами: підтримка інтеграції з ERP, CRM, GIS та іншими корпоративними системами.

Infor EAM забезпечує всеосяжний набір інструментів для ефективного управління активами, що дозволяє підвищити їх продуктивність, зменшити витрати на обслуговування та продовжити термін служби активів.

IBM Maximo — це комплексна система управління активами підприємства (EAM), яка надає організаціям можливості для ефективного управління фізичними активами протягом їхнього життєвого циклу.

Основні функції системи IBM Maximo включають:

- інвентаризація активів: ведення повного переліку активів, включаючи обладнання, транспортні засоби, інфраструктуру та інші об'єкти;
- інформація про активи: зберігання детальної інформації про активи, включаючи технічні характеристики, історію обслуговування, вартість та інші атрибути;
- планове технічне обслуговування (ПТО): планування та виконання регулярних обслуговувань для підтримки активів у робочому стані;
- графіки обслуговування: створення графіків технічного обслуговування на основі часу, показників роботи обладнання або умов експлуатації;
- інвентаризація запасів: ведення обліку запасних частин, матеріалів та витратних матеріалів;
- генерація звітів: створення стандартних та користувацьких звітів для аналізу ефективності управління технічним обслуговуванням;
- аналіз даних: Інструменти для аналізу даних з метою виявлення тенденцій, проблем та можливостей для покращення;
- API та інтеграція: інтеграція з ERP, CRM, SCADA та іншими корпоративними системами для забезпечення безперервного обміну даними;
- імпорт/експорт даних: підтримка імпорту та експорту даних для легкого перенесення інформації між системами;
- аналітика даних: використання даних з датчиків IoT та інших джерел для прогнозування потреб в обслуговуванні;
- планування на основі стану: оптимізація графіків обслуговування на основі реального стану обладнання.

IBM Maximo забезпечує широкий спектр функцій для ефективного управління активами та технічним обслуговуванням, що дозволяє підвищити

надійність і продуктивність активів, зменшити витрати на обслуговування та покращити загальну ефективність операцій.

UpKeep - це сучасна система управління технічним обслуговуванням (CMMS), яка дозволяє підприємствам автоматизувати та оптимізувати процеси технічного обслуговування та ремонту обладнання. Ця система підходить для різних галузей і масштабів бізнесу, надаючи користувачам зручні інструменти для покращення ефективності та надійності обладнання.

Основні функції UpKeep:

- створення та відстеження заявок: можливість легко створювати заявки на технічне обслуговування, призначати відповідальних осіб та відстежувати статус виконання робіт;
- планові обслуговування: створення графіків планових технічних обслуговувань для запобігання несправностей та забезпечення безперебійної роботи обладнання;
- календарі та нагадування: використання календарів для візуалізації планових робіт та отримання нагадувань про майбутні завдання;
- облік активів: ведення детального реєстру активів із зазначенням їх місцезнаходження, стану, історії ОБСЛУГОВУВАННЯ та ремонту;
- моніторинг стану активів: Регулярне відстеження технічного стану активів для своєчасного виявлення та усунення проблем;
- мобільні додатки: доступ до системи через мобільні додатки для iOS та Android, що дозволяє створювати заявки, оновлювати статуси та переглядати інформацію про активи з будь-якого місця;
- сканування штрих-кодів та QR-кодів: використання мобільних пристроїв для сканування штрих-кодів та QR-кодів активів для швидкого доступу до інформації;
- генерація звітів: створення різноманітних звітів про технічне обслуговування, використання запасів, продуктивність активів та ефективність роботи технічного персоналу.

Fiix - це комплексна система управління технічним обслуговуванням (CMMS), призначена для автоматизації та оптимізації процесів управління технічним обслуговуванням і ремонтом обладнання. Система допомагає підприємствам ефективно відстежувати, планувати та виконувати технічні роботи, забезпечуючи високу продуктивність та надійність обладнання.

Основні функції Fiix:

- створення та відстеження заявок: легке створення заявок на технічне обслуговування, можливість призначення виконавців, пріоритизація та відстеження статусу виконання;
- планові роботи: створення графіків планових технічних обслуговувань для запобігання несправностей та подовження терміну служби обладнання;
- облік активів: ведення детальної бази даних активів із інформацією про місцезнаходження, стан, історію обслуговування та ремонту;
- моніторинг стану: регулярний моніторинг стану активів та відстеження показників їх продуктивності для своєчасного виявлення та усунення проблем;
- мобільні додатки: доступ до системи через мобільні додатки для iOS та Android, що дозволяє користувачам створювати заявки, оновлювати статуси та переглядати інформацію про активи з будь-якого місця;
- сканування штрих-кодів та QR-кодів: використання мобільних пристроїв для сканування штрих-кодів та QR-кодів активів для швидкого доступу до інформації;
- генерація звітів: створення різноманітних звітів про технічне обслуговування, продуктивність активів, використання запасів та ефективність роботи технічного персоналу;
- інтеграція з іншими системами: підтримка інтеграції з ERP-системами, системами управління ланцюгами постачання та іншими корпоративними додатками для створення єдиного інформаційного простору.

eMaint CMMS (Computerized Maintenance Management System) - це комплексне програмне забезпечення для управління технічним обслуговуванням, яке допомагає організаціям ефективно планувати, відстежувати та виконувати технічне обслуговування їх активів. Система забезпечує широкий спектр функцій, що дозволяють підвищити ефективність роботи та знизити витрати на технічне обслуговування.

Основні функції eMaint CMMS:

- створення та управління заявками: легке створення заявок на технічне обслуговування, призначення виконавців та відстеження виконання робіт;
- планові обслуговування: створення та управління графіками планових технічних обслуговувань для запобігання поломок та зниження простоїв;
- календар планування: використання календаря для візуалізації та планування робіт з технічного обслуговування;
- облік активів: ведення детального обліку активів, включаючи їх розташування, стан, історію технічного обслуговування та ремонту;
- моніторинг стану активів: відстеження технічного стану активів для своєчасного виявлення та усунення проблем;
- облік запасних частин: управління запасами запасних частин та матеріалів, включаючи відстеження їх наявності та використання;
- мобільні додатки: використання мобільних додатків для доступу до системи з будь-якого місця, дозволяючи технічним працівникам створювати та оновлювати заявки, переглядати розклади та інші дані в режимі реального часу;
- сканування штрих-кодів: використання мобільних пристроїв для сканування штрих-кодів активів та запасних частин для швидкого доступу до інформації;
- генерація звітів: створення різноманітних звітів про виконання технічного обслуговування, використання запасів, продуктивність активів та інші ключові показники.

В таблиці 1 наведено порівняння деяких функцій систем конкурентів.

Таблиця 1 – Порівняння конкурентів (таблиця виконана самостійно)

Аналізований показник	Наш додаток	Infor EAM	IBM Maximo	UpKeep	Fiix	eMaint CMMS
1	2	3	4	5	6	7
Пошук приладів, які підлягають повторної перевірки	+	+	+	+	+	+
Введення статистики по перевіреним приладам	+	+	-	+	+	+
Видача необхідних звітів користувачеві	+	+	+	+	+	+
Фільтрація даних	+	+	+	+	+	+
Можливість надсилання звітів на пошту	+	-	-	+	+	+
Можливість штрих-кодування приладів	+	+	-	+	+	+
Можливість пошуку інформації в інтернеті з додатку	+	-	-	-	-	-

Як можна побачити, системи конкуренти, які знайшли широке коло застосування, в тому числі на залізничному транспорті, не мають можливості надсилання звітів на електронну адресу. Системи конкуренти, не мають можливості виходу в інтернет з додатку. В запропонованому додатку передбачена можливість надсилання звітів на електрону адресу, як в автоматичному, так і ручному режимі, а також передбачається можливість користувачам виходити в інтернет за допомогою вбудованого в додаток браузера.

1.3 Постановка задачі

У сучасних умовах розвитку залізничного транспорту важливим аспектом є забезпечення безперебійної та безпечної роботи систем залізничної автоматики. Облік приладів залізничної автоматики є критично важливим завданням, яке дозволяє забезпечити належний рівень технічного обслуговування, своєчасний ремонт та модернізацію обладнання, а також знизити ризики виникнення аварійних ситуацій. Відсутність ефективної системи обліку може призвести до нераціонального використання ресурсів, збільшення витрат на технічне обслуговування та зниження рівня безпеки руху поїздів.

В роботі необхідно спроектувати та реалізувати програмну систему обліку приладів залізничної автоматики.

Основні цілі та завдання:

а) створення ефективної системи обліку приладів залізничної автоматики:

1) розробка та впровадження програмного забезпечення для автоматизації процесу обліку;
2) забезпечення точного та своєчасного внесення даних про всі прилади залізничної автоматики;

б) ведення бази даних приладів:

1) створення централізованої бази даних, яка міститиме інформацію про всі прилади, включаючи їх технічні характеристики, місце розташування, дату встановлення, історію технічного обслуговування та ремонту;
2) забезпечення регулярного оновлення даних для підтримки їх актуальності;

в) контроль стану приладів:

1) реалізація функції моніторингу стану приладів у реальному часі;
2) виявлення несправностей та відхилень у роботі приладів з метою своєчасного реагування;

г) планування технічного обслуговування та ремонту:

1) розробка планів технічного обслуговування на основі даних про стан приладів та рекомендацій виробників;
2) автоматизація процесу формування заявок на технічне обслуговування та ремонт;

д) аналіз та звітність:

1) збір та аналіз даних про роботу приладів, технічне обслуговування та ремонти;
2) генерація звітів для керівництва з метою прийняття обґрунтованих рішень щодо модернізації та заміни обладнання.

Впровадження системи обліку приладів залізничної автоматики дозволить досягти наступних результатів:

– підвищення ефективності управління та технічного обслуговування приладів;
– зниження витрат на технічне обслуговування та ремонт за рахунок своєчасного виявлення та усунення несправностей;
– підвищення рівня безпеки руху поїздів;
– забезпечення прозорості та контролю за станом обладнання.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Для забезпечення ефективного обліку приладів залізничної автоматики необхідно розробити програмну систему, яка буде відповідати сучасним стандартам якості, надійності та безпеки. Формування вимог до такої системи включає визначення функціональних і нефункціональних вимог, які повинні бути враховані при проектуванні та реалізації програмного забезпечення.

Опишемо функціональні вимоги:

а) облік приладів залізничної автоматики:

1) ведення реєстру приладів: система повинна забезпечувати створення та ведення детальної бази даних приладів залізничної автоматики з інформацією про технічні характеристики, серійні номери, місце розташування, дати встановлення та інші важливі параметри;
2) інвентаризація: Можливість проведення регулярної інвентаризації приладів для підтвердження їх наявності та стану;

б) моніторинг і контроль стану приладів:

1) моніторинг в реальному часі: система повинна забезпечувати можливість моніторингу стану приладів у реальному часі з використанням датчиків і мережевих технологій;
2) сигналізація про несправності: автоматичне виявлення несправностей та відхилень у роботі приладів з надсиланням повідомлень відповідальним особам;

в) планування та управління технічним обслуговуванням і ремонтом:

1) планування ТО: формування графіків планового технічного обслуговування на основі даних про стан приладів та рекомендацій виробників;
2) управління заявками на ремонт: створення та обробка заявок на технічне обслуговування і ремонт, включаючи призначення відповідальних осіб та відстеження виконання робіт;

г) звітність та аналітика:

1) генерація звітів: можливість автоматичного генерування звітів про стан приладів, проведені технічні роботи, використання запасних частин та інші аспекти діяльності;
2) аналітичні інструменти: наявність інструментів для аналізу даних, що дозволяють приймати обґрунтовані рішення щодо модернізації та оптимізації технічного обслуговування.

Наведемо нефункціональні вимоги до програмної системи:

а) надійність і безпека:

1) захист даних: система повинна забезпечувати високий рівень захисту даних, включаючи шифрування інформації, контроль доступу та резервне копіювання;
2) відмовостійкість: забезпечення стабільної роботи системи навіть у випадку збоїв або відмов окремих компонентів;

б) продуктивність і масштабованість:

1) висока продуктивність: система повинна забезпечувати швидке оброблення великих обсягів даних та високий рівень відгуку на запити користувачів;
2) масштабованість: можливість розширення системи для підтримки більшої кількості приладів та користувачів без значного зниження продуктивності;

в) сумісність та інтеграція:

1) інтеграція з існуючими системами: забезпечення можливості інтеграції з іншими інформаційними системами, що використовуються на залізничному транспорті, такими як ERP-системи, системи управління технічним обслуговуванням та ремонтом (CMMS);
2) відкриті API: наявність відкритих інтерфейсів для програмування (API) для спрощення інтеграції та розширення функціоналу системи;

г) зручність обслуговування та підтримки:

1) простота встановлення та налаштування: система повинна бути легкою в установці та налаштуванні, з мінімальними вимогами до апаратного забезпечення та програмного середовища;
2) технічна підтримка: наявність кваліфікованої технічної підтримки, що забезпечує швидке вирішення проблем користувачів та регулярні оновлення системи.

д) інтуїтивно зрозумілий інтерфейс: зручний інтерфейс для користувачів різного рівня підготовки, що забезпечує легкий доступ до всіх функцій системи;

е) налаштування користувача: можливість налаштування інтерфейсу відповідно до потреб конкретного користувача або ролі.

Формування вимог до програмної системи обліку приладів залізничної автоматики є ключовим етапом у процесі її розробки. Визначення функціональних та нефункціональних вимог дозволяє створити ефективну, надійну та безпечну систему, яка відповідає сучасним стандартам і потребам залізничного транспорту. Забезпечення виконання цих вимог сприятиме підвищенню рівня безпеки та ефективності управління залізничною автоматикою, що є важливим кроком у розвитку галузі.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування програмного забезпечення

UML є стандартною мовою для візуального моделювання об'єктно-орієнтованих систем. Для розробки програмного забезпечення обліку приладів залізничної автоматики ми використовуємо UML-діаграми для відображення різних аспектів системи. Основні UML-діаграми, які будуть використані в цьому проекті, включають діаграми варіантів використання, діаграми класів, діаграми послідовності та діаграми діяльності [4].

Діаграма варіантів використання відображає взаємодію користувачів (акторів) із системою та її функціональними можливостями (варіантами використання). Use Case діаграма системи представлена на рисунку 1.

Визначимо акторів:

– адміністратор системи (старший електромеханік КВП): відповідає за налаштування, обслуговування системи та управління користувачами, а також займається додаванням, редагуванням та видаленням інформації з БД;
– користувачі системою (старші електромеханіки станцій, начальник дільниці, головний інженер дистанції): мають можливість переглядати інформацію, генерувати необхідні звіти та виводити їх на друк, пошук інформації в інтернеті з додатку.

Опишемо варіанти використання:

– управління приладами: додавання, редагування та видалення даних про прилади;
– моніторинг стану приладів: перегляд інформації про стан приладів у реальному часі;
– планування технічного обслуговування: створення графіків планового ТО;
– обробка заявок на ремонт: створення та відстеження заявок на ремонт;
– генерація звітів: створення різноманітних звітів для аналізу.

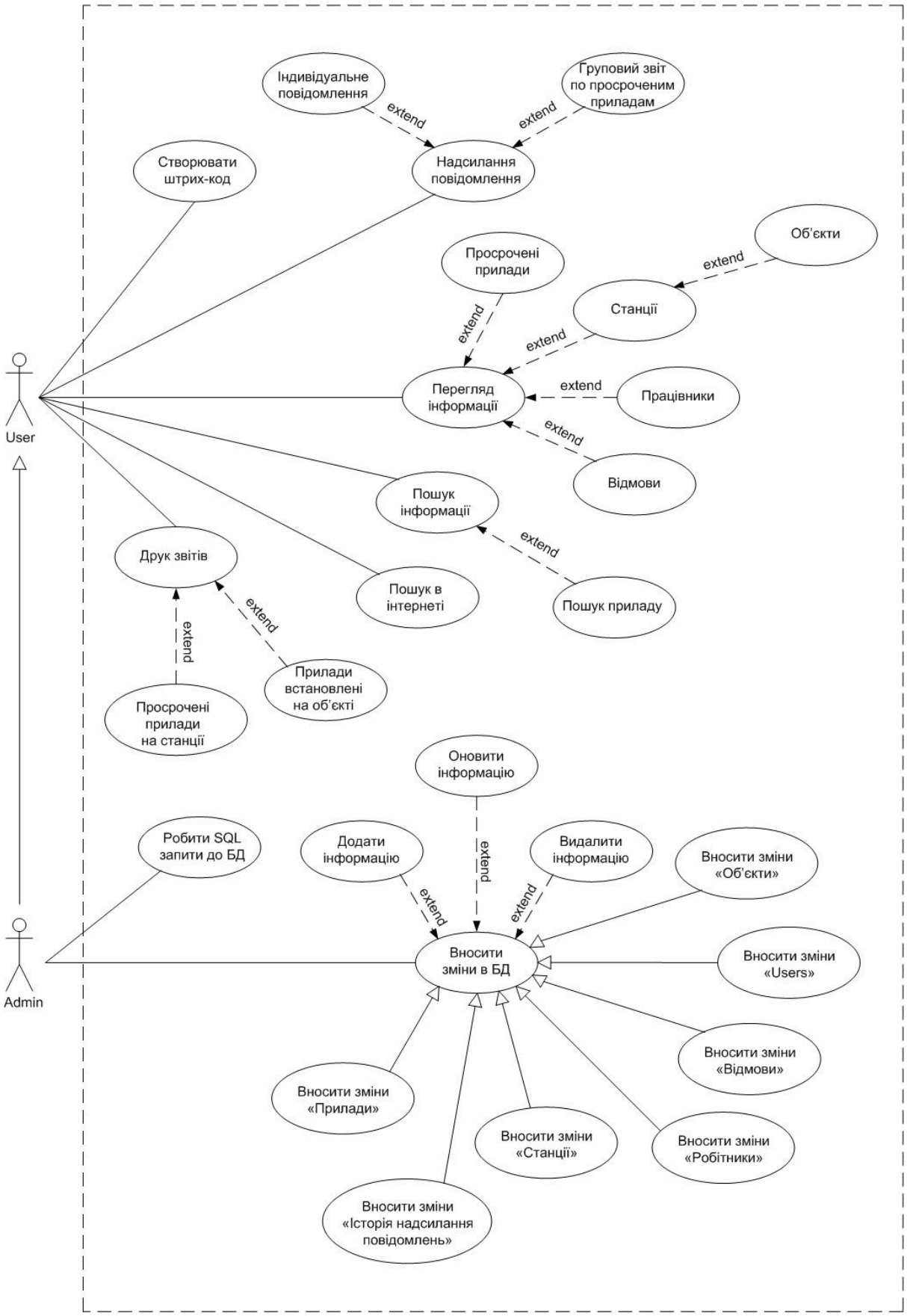


Рисунок 1 – Use Case діаграма системи (рисунок виконаний самостійно)

Діаграма послідовності (Sequence diagrams) відображає взаємодію об'єктів у системі з точки зору часу. Розглянемо відправлення повідомлення на електронну адресу за бажанням користувача (діаграма наведена на рис.2):

– система надає користувачу діалогове вікно для відправки повідомлення;
– користувач вибирає необхідний файл для відправки;
– система відображає шлях до вибраного файлу на диску;
– користувач ініціює відправлення повідомлення;
– система відправляє повідомлення на електрону адресу по протоколу SMTP.

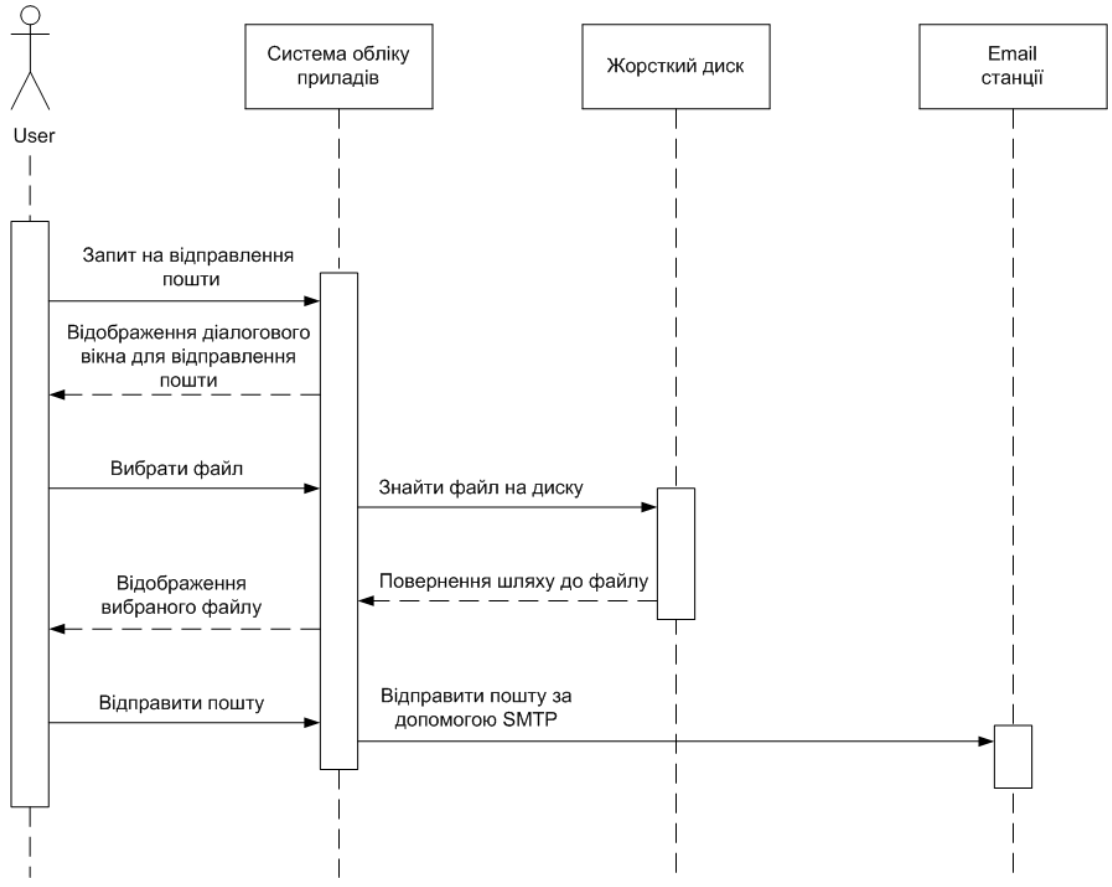


Рисунок 2 – Діаграма послідовності для відправлення повідомлення на електронну пошту (рисунок виконаний самостійно)

Перегляд інформації по приладам, які вимагають повторної перевірки:

– користувач системою хоче переглянути інформацію по приладам, які вимагають повторної перевірки на конкретній станції;

– система робить запит до БД про наявність приладів, які вимагають повторної перевірки;
– БД повертає інформацію по приладам, які вимагають повторної перевірки;
– система обробляє дані отримані з БД, та відображає їх у відповідному вікні в табличному вигляді.

Sequence діаграма представлена на рисунку 3.

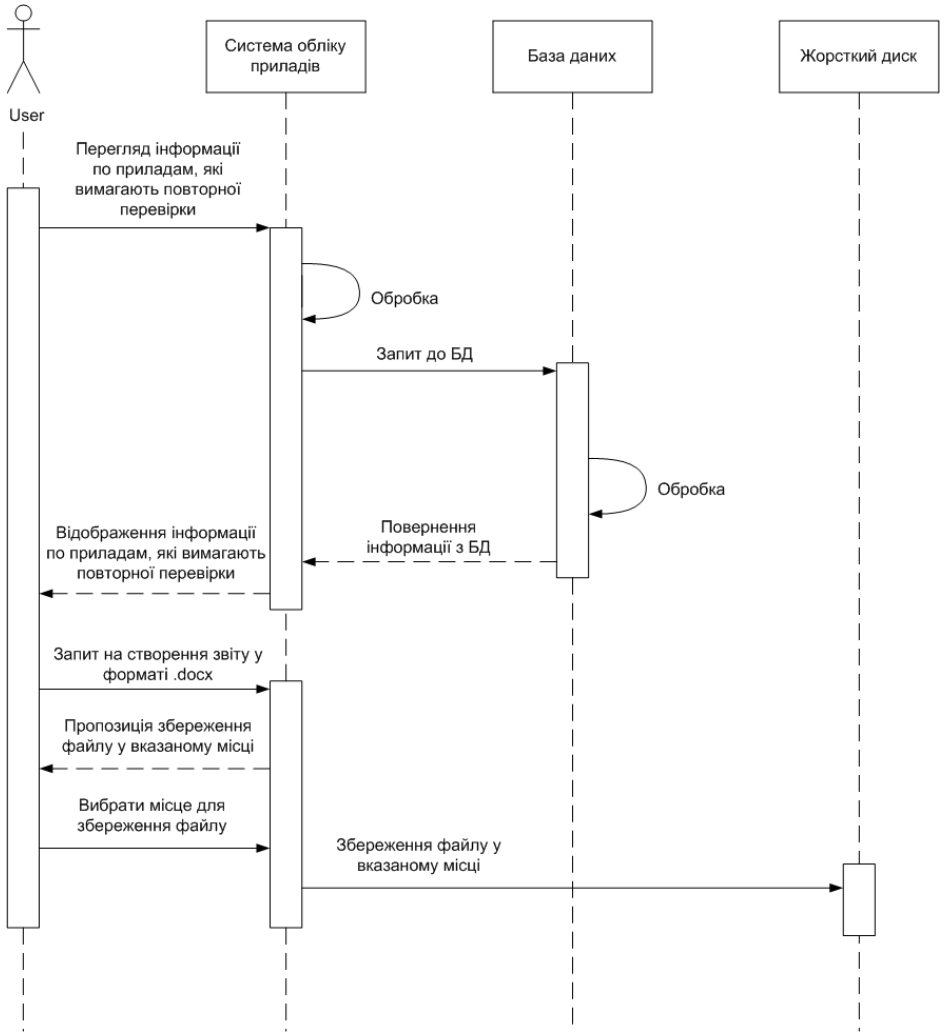


Рисунок 3 – Діаграма послідовності для перегляду інформації по приладам, які вимагають повторної перевірки (рисунок виконаний самостійно)

Діаграма діяльності відображає робочі процеси та логіку виконання завдань у системі. Діаграма, що демонструє процес автоматичної відправки звітів на станції, по приладам, які вимагають повторної перевірки наведено на рис.4.

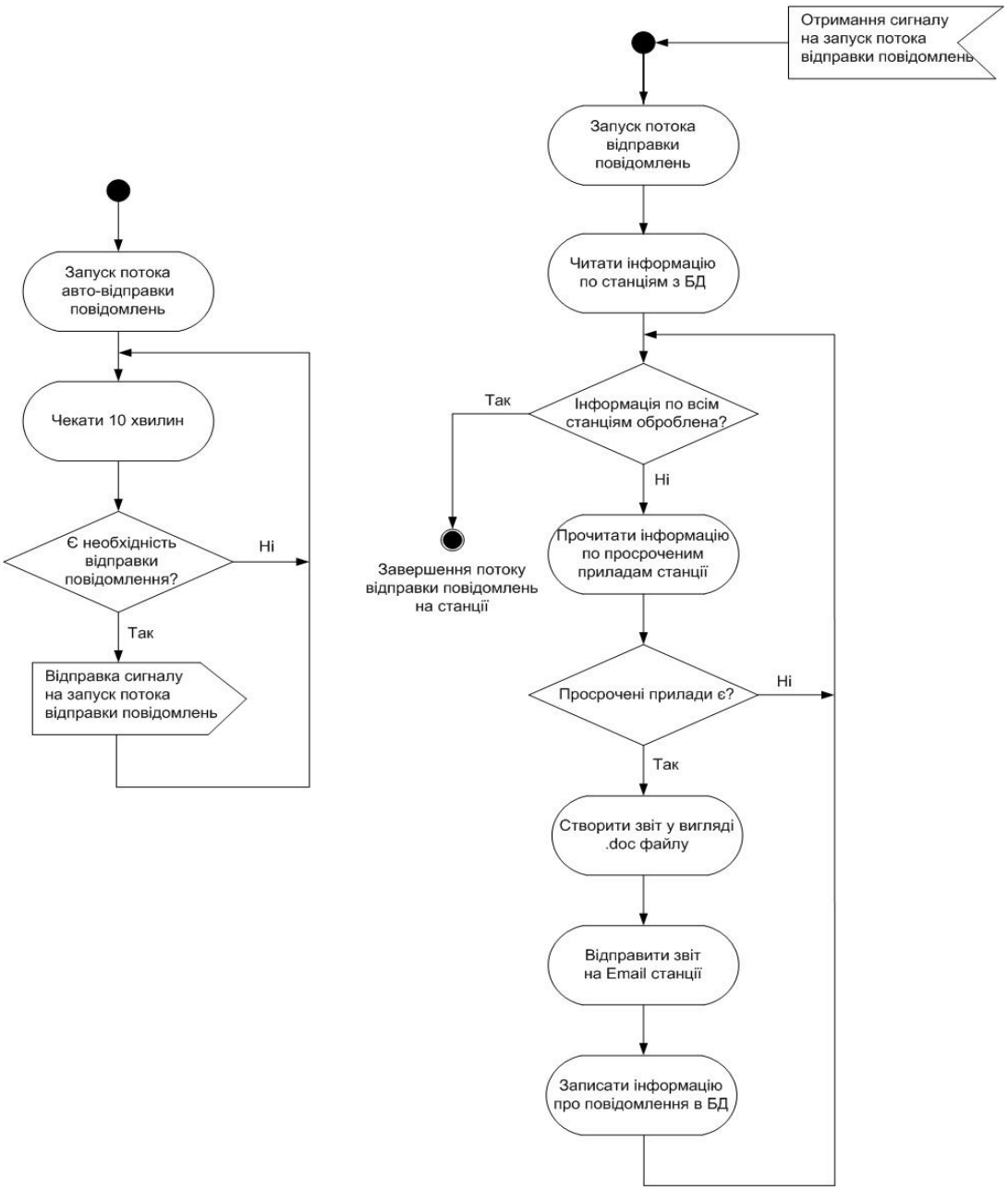


Рисунок 4 – Activity діаграма процесу автоматичної відправки звітів на електрону адресу станції (рисунок виконаний самостійно)

Опишемо основні кроки:

– ініціація: система створює потік для відправки звітів на станції;
– підготовка: система перевіряє наявність приладів, які вимагають повторної перевірки по кожній станції та створює звіт у форматі .docx;
– виконання: відправка звіту на електрону адресу станції по протоколу SMTP;

– завершення: система вносить інформацію до БД про відправлене повідомлення.

Діаграма активностей, яка демонструє процес входу в систему представлена на рисунку 5.

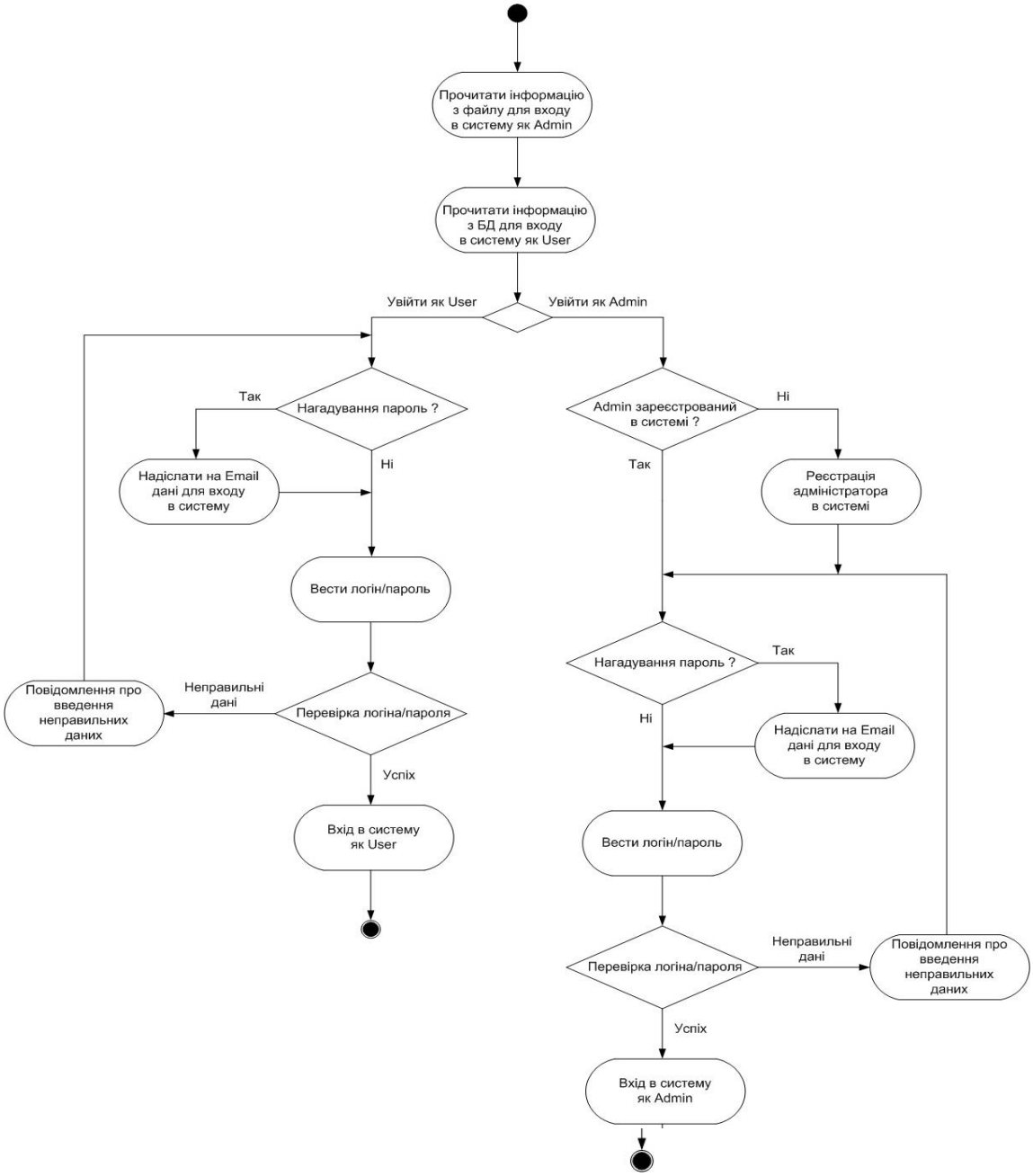


Рисунок 5 – Activity діаграма процесу входу в систему (рисунок виконаний самостійно)

Основні кроки для входу в систему:

– ініціація: користувач запускає додаток DataBaseKIP.exe;

– підготовка: користувач вибирає режим входу до системи User/Admin та вводить логін та пароль. У разі, якщо користувач системою забув логін чи пароль, на електрону адресу користувача надсилається повідомлення з нагадуванням логіна та пароля;
– виконання: система перевіряє правильність вводу логіна та пароля;
– завершення: користувач входить в систему як User чи Admin.

Використання UML для проектування програмного забезпечення обліку приладів залізничної автоматики забезпечує ясне та структуроване розуміння системи, її компонентів та взаємодії між ними. UML-діаграми допомагають не лише розробникам, але й іншим зацікавленим сторонам, таким як користувачі та менеджери, зрозуміти структуру та функціональність системи, що сприяє успішній реалізації проекту.

3.2 Проектування архітектури програмного забезпечення

Проектування архітектури програмного забезпечення (ПЗ) для обліку приладів залізничної автоматики є важливим етапом у розробці системи, яка повинна бути надійною, масштабованою та легкою в обслуговуванні. Архітектура системи визначає її структуру, основні компоненти та взаємодію між ними. У цьому розділі розглянемо ключові аспекти проектування архітектури ПЗ, включаючи вибір архітектурного стилю, визначення основних модулів та компонентів, а також забезпечення нефункціональних вимог.

Для системи обліку приладів залізничної автоматики доцільно обрати багаторівневу (багатошарову) архітектуру.

Багаторівнева архітектура клієнт-сервер (іноді відома як багатошарова архітектура) є одним із найпоширеніших підходів до побудови інформаційних систем. Вона розділяє систему на кілька рівнів (шарів), кожен з яких виконує певні функції, що дозволяє зменшити складність розробки та покращити масштабованість і підтримуваність системи. Основні рівні багаторівневої архітектури клієнт-сервер:

Рівень клієнта (Client Layer) є інтерфейсом користувача і відповідає за взаємодію з користувачем. Він може бути реалізований як настільний додаток, мобільний додаток або веб-додаток, який запускається у браузері. Основні функції цього рівня включають:

- відображення даних, отриманих від сервера;
- забезпечення введення даних користувачем;
- передача запитів на сервер та отримання відповідей.

Рівень презентації (Presentation Layer) відповідає за обробку та форматування даних для їх відображення на рівні клієнта. Це може включати:

- форматування даних для користувача;
- перетворення даних між різними форматами (наприклад, XML або JSON);
- валідація введених даних перед їх передачею на наступні рівні.

Логічний рівень (Business Logic Layer) або рівень бізнес-логіки відповідає за обробку запитів та реалізацію бізнес-правил. Основні функції цього рівня включають:

- обробка запитів, що надходять від рівня презентації;
- застосування бізнес-правил та логіки;
- взаємодія з рівнем даних для отримання або зберігання даних.

Рівень доступу до даних (Data Access Layer) відповідає за взаємодію з базою даних або іншими джерелами даних. Основні функції цього рівня включають:

- формування запитів до бази даних;
- виконання запитів та отримання результатів;
- перетворення результатів запитів у формат, придатний для логічного рівня.

Рівень зберігання даних (Data Storage Layer) включає базу даних або інші механізми зберігання даних. Основні функції цього рівня включають:

- зберігання даних у відповідній структурі (реляційні бази даних MSQL);
- забезпечення доступності та цілісності даних;
- відновлення даних на запит від рівня доступу до даних.

Багаторівнева архітектура клієнт-сервер є потужним підходом до побудови складних інформаційних систем, що дозволяє забезпечити гнучкість, масштабованість та зручність в обслуговуванні. Вона широко використовується в сучасних корпоративних додатках для забезпечення ефективного управління та обробки даних.

3.3 Проектування структури зберігання даних

Проектування структури зберігання даних є важливим етапом розробки системи обліку приладів залізничної автоматики. Ефективна структура бази даних забезпечує зручне зберігання, швидкий доступ до інформації та легке управління даними [6]. В даному розділі розглянемо основні принципи та етапи проектування структури зберігання даних, а також ключові таблиці та взаємозв'язки між ними.

Основні принципи проектування структури зберігання даних:

- нормалізація: використання процесу нормалізації для усунення надлишкових даних та запобігання аномаліям під час вставки, оновлення та видалення даних;
- цілісність даних: забезпечення цілісності даних за допомогою первинних та зовнішніх ключів, а також обмежень цілісності;
- продуктивність: оптимізація структури бази даних для забезпечення високої продуктивності запитів;
- масштабованість: проєктування структури даних з урахуванням можливості майбутнього розширення системи;
- безпека: забезпечення захисту конфіденційних даних через контроль доступу та шифрування.

На підставі аналіза предметної області побудуємо ER-діаграму (див. рис. 6).

ER-діаграма (діаграма "сутність-зв'язок") є інструментом для моделювання даних, який використовується для представлення логічної структури бази даних. Вона описує дані та їх взаємозв'язки, допомагаючи зрозуміти, як дані пов'язані один з одним у системі. ER-діаграми зазвичай використовують на етапі проектування баз даних.

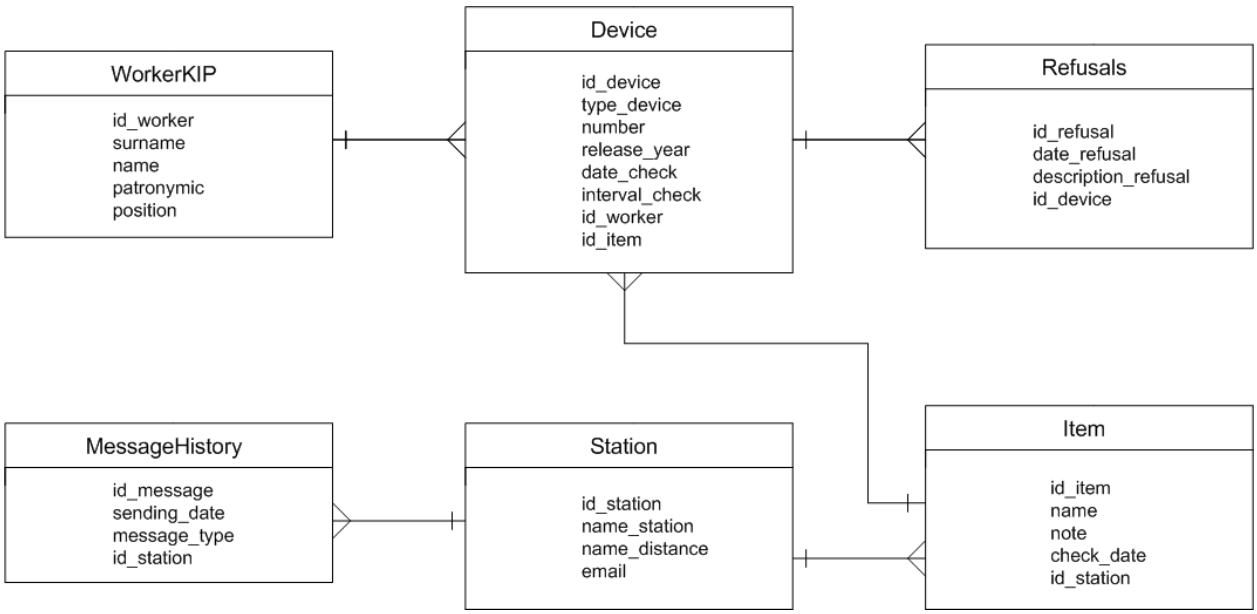


Рисунок 6 – ER діаграма системи (рисунок виконаний самостійно)

Для створення схеми реляційної бази даних використаємо створену ER-діаграму, сутності якої розглянемо як таблиці БД з відповідними атрибутами.

Опишемо основні таблиці БД для системи обліку приладів залізничної автоматики та взаємозв'язки між ними.

Таблиця "Працівники КВП" (WorkerKIP):

– id_worker - унікальний ідентифікатор працівника, первинний ключ;
– surname – прізвище працівника;
– name – ім’я працівника;
– patronymic – по-батькові працівника;
– position – посада працівника.

Таблиця "Прилади" (Device):

– id_device - унікальний ідентифікатор приладу, первинний ключ;
– type_divice - тип приладу;
– number - номер приладу;
– release_year - рік виготовлення приладу;
– date_check – дата перевірки приладу;
– interval_check – періодичність перевірки приладу;

– id_worker - ідентифікатор працівника, який перевіряв прилад, зовнішній ключ;
– id_item - ідентифікатор об’єкта, на якому встановлений прилад, зовнішній ключ.

Таблиця "Відмови" (Refusals):

– id_refusal - унікальний ідентифікатор відмови, первинний ключ;
– date_refusal – дата відмови;
– description_refusal – опис відмови;
– id_device –ідентифікатор приладу, зовнішній ключ.

Таблиця "Об’єкт" (Item):

– id_item - унікальний ідентифікатор об’єкта, первинний ключ;
– name – назва об’єкта;
– note – примітка;
– check_date – дата ревізії об’єкта;
– id_station - ідентифікатор станції, зовнішній ключ.

Таблиця "Стація" (Station):

– id_station - унікальний ідентифікатор станції, первинний ключ;
– name_station – назва станції;
– name_distance – назва дистанції;
– email – електронна адреса станції.

Таблиця "Історія надсилання повідомлень" (MessageHistory):

– id_message - унікальний ідентифікатор повідомлення, первинний ключ;
– sending_date – дата відправки повідомлення;
– message_type – тип повідомлення;
– id_station - ідентифікатор станції, зовнішній ключ.

Також створимо таблицю "Користувачі" Users:

– id_user (PK) - унікальний ідентифікатор користувача, первинний ключ;
– login – логін користувача;
– password – пароль користувача;
– email – електронна адреса користувача.

На рисунку 7 приведена схема бази даних.

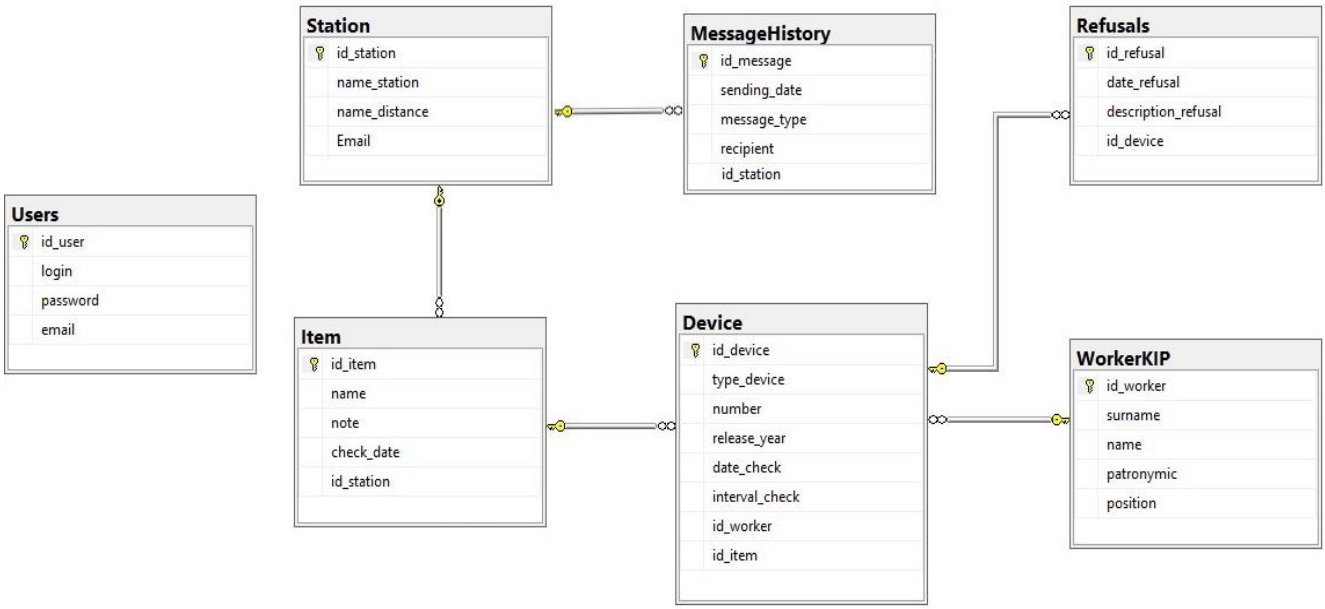


Рисунок 7 – Схема бази даних (рисунок виконаний самостійно)

3.4 Приклади найцікавіших алгоритмів та методів

У системах обліку приладів залізничної автоматики застосовуються різноманітні алгоритми та методи для забезпечення надійної роботи, ефективного обслуговування та управління даними. Нижче розглянуто кілька найцікавіших алгоритмів та методів, що використовуються в таких системах

Розглянемо алгоритм створення штрих-коду, який дозволяє створити штрих код EAN-13, який дозволяє за допомогою сканера штрих-коду зчитувати інформацію про прилади.

Ключові етапи алгоритму:

– пошук приладу для якого необхідно створити штрих-код;

– розрахунок контрольної суми штрих коду;

– генерація штрих коду за допомогою бібліотеки BarcodeLib;

– збереження штрих-коду у вказаному місці у форматі PNG.

Також заслуговує уваги реалізація методу відправки повідомлення. Цей метод дозволяє відправляти повідомлення на відповідні станції по протоколу SMTP.

Метод відправки повідомлення має наступне визначення:

```
public static void SendMail(string mailRecipient, string nameSender, string subject, string text, string attachmentPath);
```

Метод приймає наступні дані:

– mailRecipient – електрона адреса отримувача;
– nameSender – ім’я відправника пошти;
– subject – тема повідомлення;
– text – текст повідомлення;
– attachmentPath – шлях до файлу, який додається до повідомлення (за необхідністю).

Ключові етапи методу:

– встановлюємо Email відправника повідомлення, та його ім’я;
– встановлюємо Email отримувача повідомлення;
– встановлюємо текст повідомлення та тему;
– за необхідністю додаємо файл до повідомлення;
– за допомогою SMTP сервера відправляємо повідомлення;
– звільнюємо всі ресурси пов’язані з відправкою повідомлення.

Алгоритм створення звіту по простроченим приладам на станції (рис. 8) дозволяє створювати звіт по простроченим приладам на станції у форматі .docx.

Ключові етапи алгоритму:

– вказати місце де необхідно зберегти файл, та його ім’я;
– за наявністю просрочених приладів, створюємо документ за допомогою бібліотеки Microsoft.Office.Interop.Word;
– до документа додаємо Header та Footer з необхідною інформацією;
– до документа додаємо таблицю, яку заповнюємо інформацією по простроченим приладам;
– зберігаємо файл у вказаному місці у форматі .docx.

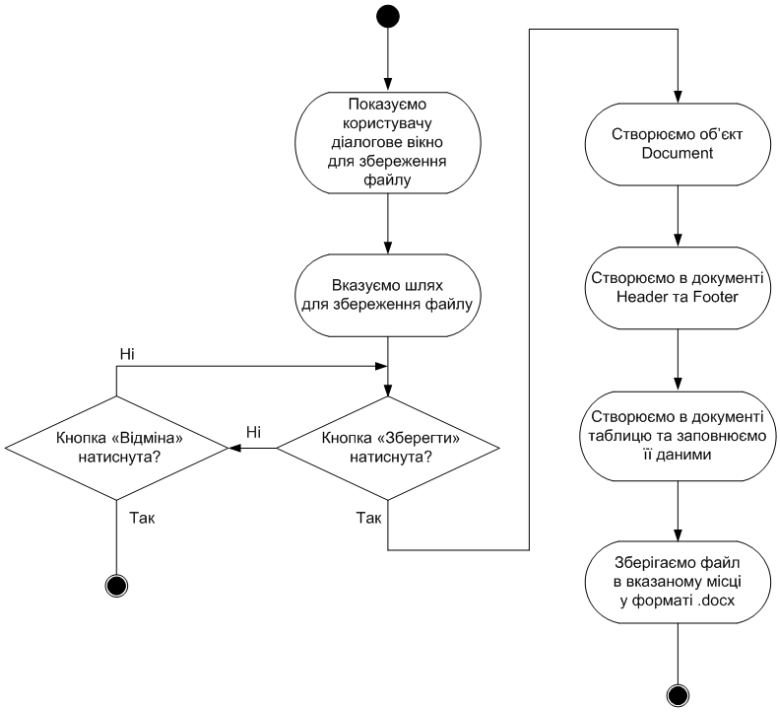


Рисунок 8 - Алгоритм створення звіту по простроченим приладам на станції (рисунок виконаний самостійно)

Алгоритм відправки звіту по приладам, які вимагають повторної перевірки на відповідну станцію дозволяє відправляти звіт по простроченим приладам на відповідну станцію у форматі .docx по протоколу SMTP.

Ключові етапи алгоритму:

– отримуємо з бази даних інформацію по протермінованим приладам на відповідній станції;
– за наявністю протермінованим приладів, створюємо документ за допомогою бібліотеки Microsoft.Office.Interop.Word та зберігаємо його у тимчасовій папці Temp у форматі .docx;
– відправляємо звіт на Email станції за допомогою протоколу SMTP;
– до бази даних заносимо інформацію про відправлене повідомлення;
– очищаємо папку Temp.

Застосування різноманітних алгоритмів та методів у системах обліку приладів залізничної автоматики дозволяє забезпечити ефективне управління даними, своєчасне виявлення та усунення несправностей, оптимізацію технічного

обслуговування. Це сприяє підвищенню надійності та безпеки залізничної інфраструктури, а також зменшенню витрат на її обслуговування.

3.5 Створення UI/UX для системи обліку приладів залізничної автоматики

Розробка інтерфейсу користувача (UI) та загального користувацького досвіду (UX) є ключовими компонентами для забезпечення ефективності та зручності використання системи обліку приладів залізничної автоматики. Успішний дизайн UI/UX дозволяє спростити взаємодію користувачів з системою, зменшити кількість помилок та покращити загальну продуктивність. У цьому розділі розглянемо основні етапи та принципи створення UI/UX для даної системи.

Основні принципи UI/UX дизайну:

- простота та інтуїтивність: інтерфейс повинен бути максимально зрозумілим та простим у використанні, щоб користувачі могли швидко знаходити необхідні функції;
- консистентність: всі елементи інтерфейсу повинні бути стилістично узгодженими та мати однакову поведінку на всіх сторінках системи;
- доступність: інтерфейс має бути доступним для користувачів з різними можливостями, включаючи людей з обмеженими можливостями;
- зворотній зв'язок: система повинна надавати зворотний зв’язок на дії користувача, щоб той знав про успішне виконання операцій або про помилки;
- адаптивність: інтерфейс повинен добре працювати на різних пристроях та екранах різних розмірів, забезпечуючи однаковий досвід для всіх користувачів.

Основні компоненти інтерфейсу складаються з головного вікна програми, вікна детальної інформації по станціям, відправлення пошти, створення штрих-коду, вікон перегляду деательної інформації по обїєктам та приладам. Розглянемо детальніше кожне з них.

Головне вікно програми відображає таблицю в якій присутня інформація по всім приладам, включаючи їх місцезнаходження. Прилади, які вимагають

повторної перевірки виділені червоним кольором. Також в головному вікні, у верхній частині знаходиться панель меню з такими пунктами (див. рис. 9):

- «БД» (призначена для внесення, редагування, видалення певної інформації з бази даних. Доступна лише адміністратору);
- «Робота з БД» (призначена для взаємодії з БД за допомогою SQL запитів. Доступна лише адміністратору);
- «Детальна інформація» (призначена для відображення користувачеві інформації по станціям, працівникам, відмовам);
- «Просрочені прилади» (призначена для відображення користувачеві інформації по приладам, які вимагають повторної перевірки);
- «Пошук»: (призначена для пошуку приладів за певними критеріями);
- «Пошта»: (призначена для відправлення пошти на електрону адресу станції, а також для налаштування періодичності відправки звітів по просроченим приладам в автоматичному режимі. Адміністратору доступна можливість переглядати історію надсилань повідомлень системою);
- «Інтернет»: (призначена для пошуку інформації в інтернеті за допомогою власного веб-браузера).

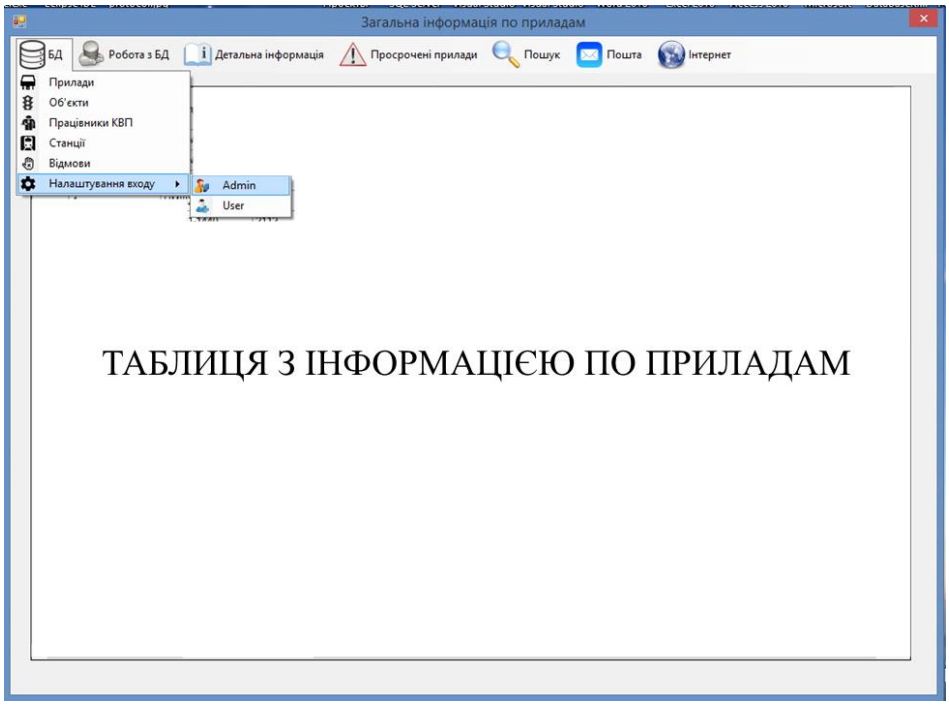


Рисунок 9 - Головне вікно програми

Вікно детальної інформації по станціям (див. рис. 10) дозволяє:

– переглядати інформацію по об’єктам, які знаходяться на станції;
– переглядати інформацію по приладам, які вимагають повторної перевірки;
– переглядати інформацію по загальній кількості приладів певного типу;
– переглядати інформацію по загальній кількості приладів певного типу, які вимагають повторної перевірки.

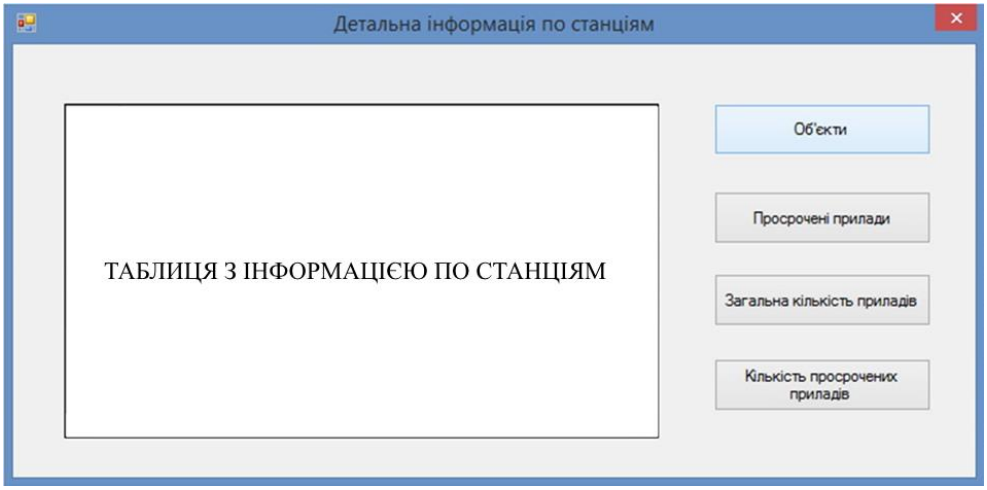


Рисунок 10 - Вікно детальної інформації по станціям

Вікно відправлення пошти (див.рис.11) призначене для відправлення пошти на електрону адресу станції. Користувач у відповідні поля вказує: одержувача, відправника, тему повідомлення, текст повідомлення, та за необхідністю додає до повідомлення файл.

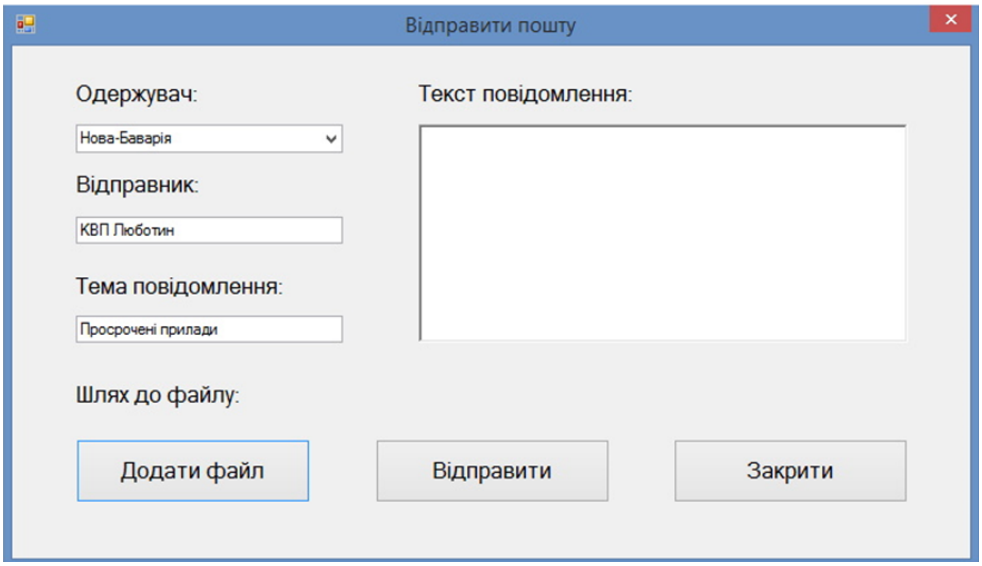


Рисунок 11 - Вікно відправлення повідомлень

Вікно створення штрих коду (див. рис.12) призначене для створення штрих коду на відповідний прилад, а також для збереження його у вказаному місці у форматі .png.

Рисунок 12 - Вікно створення штрих коду

Вікно детальної інформації по об’єктах (див.рис.13) дозволяє:

– переглядати інформацію по приладам, які знаходяться на об’єкті;
– створювати звіт у форматі .docx по об’єктам, які знаходяться на станції.

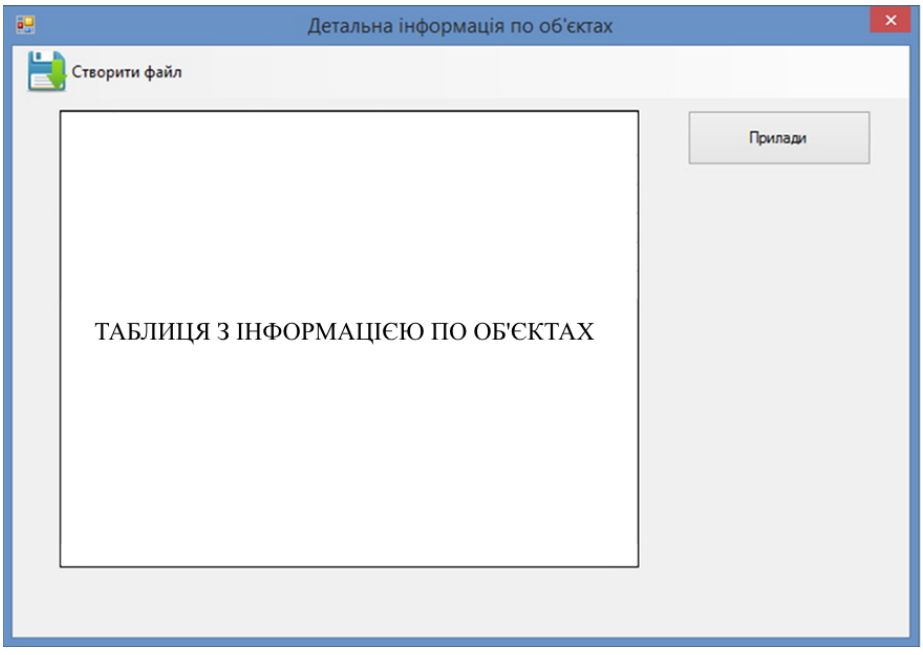


Рисунок 13 - Вікно детальної інформації по об’єктах

Вікно детальної інформації по приладам (див.рис.14) дозволяє переглядати інформацію по приладам, які знаходяться на об’єкті, та виводити її на друк.

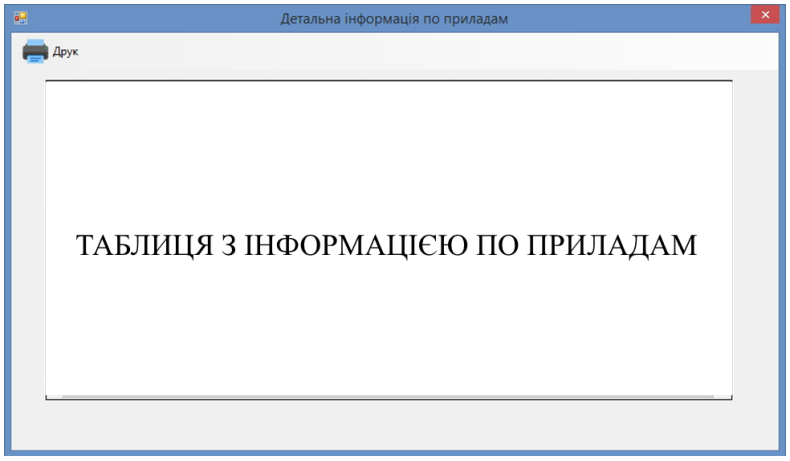


Рисунок 14 - Вікно детальної інформації по приладам

Вікно входу в систему (див.рис.15) призначене для реєстрації та авторизації користувачів, а також для нагадування логіна та пароля.

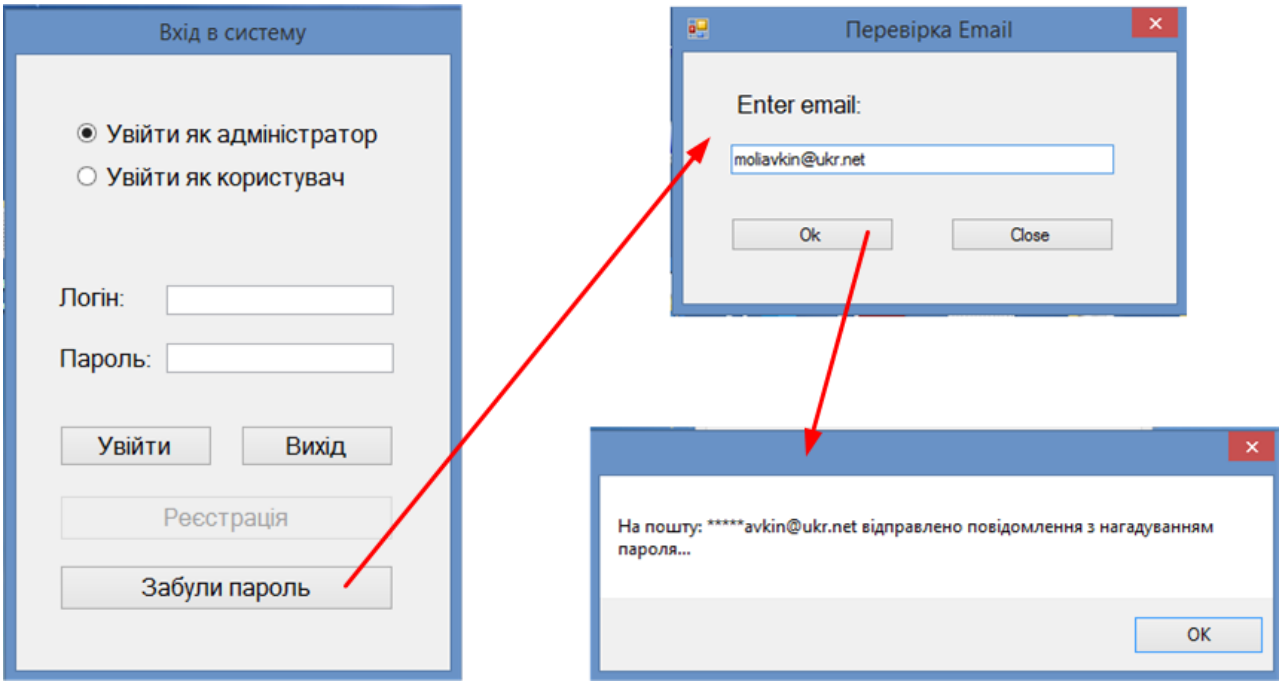


Рисунок 15 - Вікно входу в систему

Процес створення UI/UX дизайну для системи обліку приладів залізничної автоматики включає декілька важливих етапів, від дослідження та аналізу потреб користувачів до розробки та тестування інтерактивних прототипів. Врахування основних принципів дизайну та інтерактивна взаємодія з користувачами на кожному етапі розробки дозволяє створити зручний та ефективний інтерфейс, що сприяє підвищенню продуктивності та задоволеності користувачів.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

При розробці ІС «Облік приладів залізничної автоматики» були використані наступні технології:

- мова програмування C#;
- технологія Windows Forms для розробки графічного інтерфейсу;
- фреймворк ADO.NET, для взаємодії додатку з базою даних;
- система управління базами даних (СУБД) Microsoft SQL Server, для зберігання даних;
- бібліотека CefSharp.WinForms, для розробки власного web браузера на основі движка Chromium.

При розробці програмної системи було використано різноманітні програмні підходи для досягнення поставленої задачі.

Нижче наведено декілька реалізацій функцій системи, які булу використано в додатку.

4.1 Реалізація функціоналу авто-відправлення повідомлення

При запуску програмної системи визивається статичний метод AutoCallThread класу ThreadMail. В цей метод передається об’єкт класу ProgramStatus, який зберігає таку інформацію як: дата і час останьої відправки повідомлення, періодичність відправки повідомлення, режим відправки повідомлення.

В методі AutoCallThread запускається потік, який існує на весь час роботи програмної системи. При завершені роботи програмної системи, цей потік ліквідується. Робота потоку реалізована в методі AutoCall. В цьому методі існує нескінчений цикл, в якому через кожні 10 хвилин, перевіряється необхідність відправки звіту по просроченим приладам, на електрону пошту відповідної станції.

Якщо поточна дата і час, більша за дату і час останьої відправки повідомлення + періодичність відправки, то в методі RunThread класу ThreadMail

запускається потік відправки повідомлень, при умові, що включений режим авто-відправки повідомлень.

Робота потоку реалізована в методі Send. Цей метод приймає логічний параметр, який визначає режим відправки повідомлення (true - автоматичний, false - ручний). В цьому методі відбувається аналіз даних на необхідність відправки звітів. Аналіз даних відбувається в методі DataAnalysis класу Station. При роботі потоку блокується пункт меню «Відправити звіт на всі станції».

Наведемо код функції відправлення повідомлення:

```
public static void AutoCallThread(ProgramStatus PrSt)
{
  if (!flagAutoCall && PrSt != null)
  {
    ps = PrSt;
    flagAutoCall = true;
    StaticDataClass.listThread.Add(myAutoCallThread);
    myAutoCallThread.Start();
  }
}
// Функція авто-відправки повідомлення
private static void AutoCall()
{
  while (true)
  {
    ps.SendMessage(true);
    for (int i = 0; i < 120; i++) Thread.Sleep(5000);
  }
}
public static void RunThread(bool flag)
{
  if (myThread.ThreadState == ThreadState.Unstarted)
  {
    StaticDataClass.listThread.Add(myThread);
    myThread.Start(flag);
  }
  else if (myThread.ThreadState == ThreadState.Stopped)
  {
    myThread = new Thread(Send);
    StaticDataClass.listThread.Add(myThread);
    myThread.Start(flag);
  }
}
private static void Send(object objg)
{
  bool flag = (bool)objg;
  try
  {
    StaticDataClass.sendToolStripMenuItem.Enabled = false;
  }
  catch (Exception ex) {}
  Station.DataAnalysis(flag);
```

```
        try
        {
          StaticDataClass.sendToolStripMenuItem.Enabled = true;
        }
        catch (Exception ex) {}
    }
}
```

4.2 Реалізація функціоналу відправки повідомлення на електрону пошту

В метод SendMail класу SendMailClass передаються такі аргументи, як:

- електрона адреса отримувача;
- ім’я відправника;
- тема повідомлення;
- текст повідомлення;
- шлях до файлу, який необхідно вкласти у повідомлення.

За допомогою класу MailMessage створюємо об’єкт повідомлення.

За допомогою класу Attachment робимо вкладення у повідомлення.

За допомогою класу SmtpClient створюємо об’єкт, за допомогою якого зможемо відправляти повідомлення по протоколу SMTP. Повідомлення відправляємо за допомогою методу Send.

На рисунку 16 наведений вміст поштової скриньки після надсилання повідомлення з додатку.

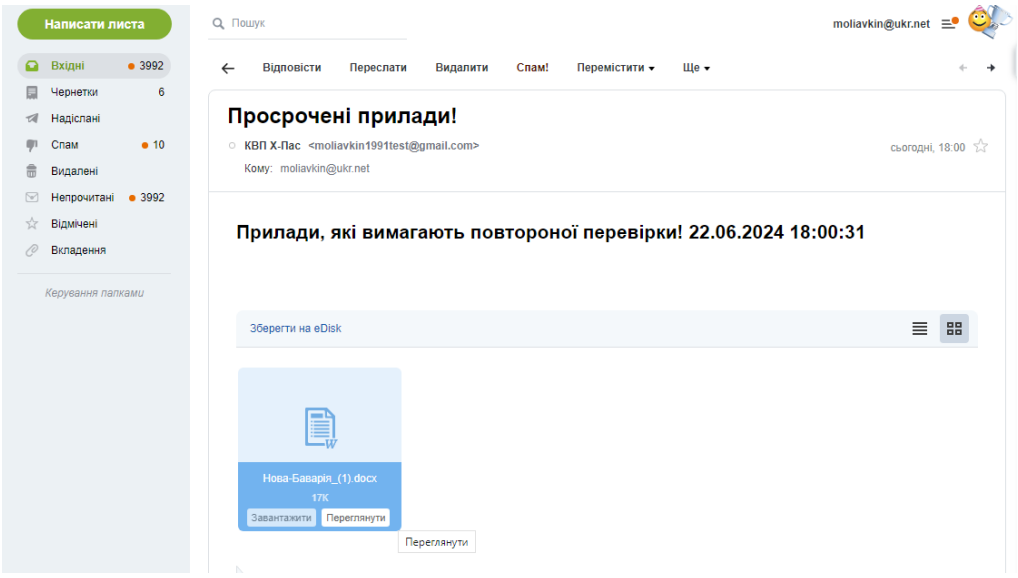


Рисунок 16 – Вміст поштової скриньки після надсилання повідомлення з додатку (рисунок виконаний самостійно)

Наведемо код функції відправлення повідомлення:

```
  public static void SendMail(string MailRecipient, string NameSender,
string
  Subject, string text, string AttachmentPath)
  {
    Attachment attachment = null;
    MailAddress from = new MailAddress(MailSender, NameSender);
    MailAddress to = new MailAddress(MailRecipient);
    MailMessage m = new MailMessage(from, to);
    if (File.Exists(AttachmentPath))
    {
      attachment = new Attachment(AttachmentPath);
      m.Attachments.Add(attachment);
    }
    m.Subject = Subject;
    text = "<h1>" + text + "</h1>";
    m.Body = text;
    m.IsBodyHtml = true;
    SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
    smtp.Credentials = new NetworkCredential(MailSender, password);
    smtp.EnableSsl = true;
    smtp.Send(m);
    if (attachment != null) attachment.Dispose();
    m.Dispose();
    smtp.Dispose();
  }
```

4.3 Реалізація функціоналу створення звіту

Документ у форматі .docx створюється за допомогою бібліотеки Microsoft.Office.Interop.Word.

В методі GetWordFile створюємо об’єкт класу Application та Document, які використовуються при роботі з Word файлом.

На рисунку 17 представлений зовнішній вигляд оформлення звіту з відповідною інформацією.

В документ додаємо верхній та нижній колонтитул, необхідний текст, таблицю. До таблиці додаємо назву, та заповнюємо її необхідною інформацією.

Після заповнення документа, вказуємо його ім’я та зберігаємо його у форматі .docx у вказаному місці за допомогою метода SaveAs2.

Закриваємо документ, та звільнюємо ресурси пов’язані з об’єктом word класу Application.

Дата створення документа: 22.06.2024 18:00:30

Станція: Нова-Баварія

Прилади, які вимагають перевірки

ID	Тип приладу	Номер приладу	Рік випуску	Дата перевірки	Періодичність перевірки (міс)	Перевірив	Назва об'єкта
7	ТШ-65В 2М	56	2006	27.06.2021	12	Дудін	Релейна шафа Нн
8	ИМШ1-1700	346	1968	26.07.2021	12	Дудін	Релейна шафа Нн
9	БИ-ДА	67678	2007	20.08.2021	12	Дудін	Релейна шафа Нн
33	ТШ-65В 2М	32	2006	13.07.2022	12	Дудін	Релейна шафа Нн
61	ИМВШ-110	6589	1982	22.06.2023	12	Нікітін	Статив 153

Рисунок 17 – Зовнішній вигляд оформлення звіту (рисунок виконаний самостійно)

Наведемо код функції створення звіту:

```
private void GetWordFile()
{
    try
    {
      Application word = new Application();
      object missing = Missing.Value;
      Document doc = word.Documents.Add(ref missing, ref missing, ref
      missing, ref missing);
      // код створення верхнього та нижнього колонтитулу
      // ..................................................
      Table firstTable = doc.Tables.Add(para1.Range, dt.Rows.Count + 1,
      dt.Columns.Count, ref missing, ref missing);
      firstTable.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;
      firstTable.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;

      for (int i = 0; i < dt.Columns.Count; i++)
      {
         firstTable.Cell(1, i + 1).Range.Text = dt.Columns[i].ColumnName;
      }
      for (int i = 0; i < dt.Rows.Count; i++)
      {
        for (int j = 0; j < dt.Columns.Count; j++)
        {
          if (j == 4)
          {
            string dateCheck = dt.Rows[i][j].ToString();
            dateCheck = dateCheck.Substring(0, dateCheck.IndexOf(' '));
            firstTable.Cell(i + 2, j + 1).Range.Text = dateCheck;
          }
```

```
            else
            {
              firstTable.Cell(i + 2, j + 1).Range.Text =
              dt.Rows[i][j].ToString();
            }
          }
        }
        count++;
        string pathFile = string.Format("{0}_({1}).docx", Name, count);
        PathSend = path + pathFile;
        doc.SaveAs2(PathSend);
        doc.Close(ref missing, ref missing, ref missing);
        word.Quit(ref missing, ref missing, ref missing);
      }
      catch (Exception ex) {}
}
```

4.4 Реалізація функціоналу створення штрих коду EAN13

Для створення штрих-коду необхідно спочатку згенерувати цифровий код, структура якого наведена на рис.18, а потім – графічний код з використанням об’єкта класу Barcode в методі Encode.

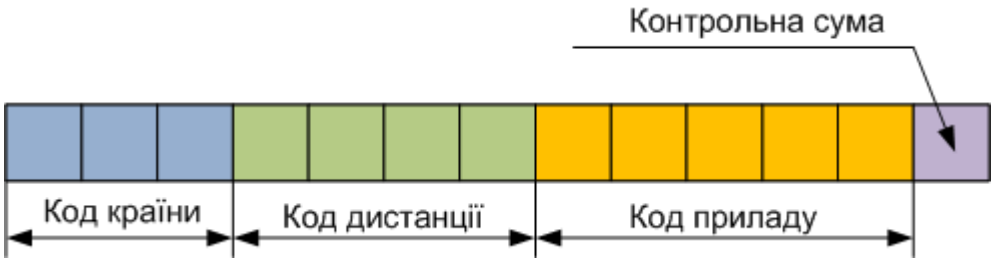


Рисунок 18 – Структура штрих-коду EAN13 (рисунок виконано самостійно)

Процедура генерації 13-ти значного коду EAN13 відбувається наступним чином:

- в метод Convert, передається унікальний ідентифікатор приладу;
- в коді EAN13 ідентифікатор приладу 5-ти значний, тому в методі Convert за необхідністю перед ID приладу додається певна кількість нулів;
- до 5-ти значного ідентифікатору приладу в коді EAN13 також додається 3-х значний код країни, 4-х значний код дистанції, та контрольна сума.

Наведемо код функції генерації штрих-коду у вигляді 13-ти значного коду:

```
private string Convert(string id)
  {
    if (id.Length > 5) return null;
```

```
    string codeCountry = "482";          // код країни
    string codeDistance = "6595";        // код дистанції
    char[] mask = { '0', '0', '0', '0', '0' };
    for (int i = mask.Length - id.Length, j = 0; i < mask.Length; i++, j++)
    {
      mask[i] = id[j];
    }
    string text = codeCountry + codeDistance + new string(mask);
    text += Control(text);
    labelCode.Text = "Код: " + text;
    return text;
  }
```

Контрольна сума визначається в методі Control наступним чином:

- метод приймає 12-ти значний код (без контрольної суми);
- складаємо цифри, що стоять на парних позиціях, потім на непарних позиціях;
- складаємо результат складання цифр на парних позиціях, помножений на три та результат складання цифр на непарних позиціях;
- контрольне число є різницею між остаточною сумою і найближчим до неї найбільшим числом, кратним десяти.

Наведемо код функції визначення контрольної суми:

```
private char Control(string text)
  {
    int sum1 = 0, sum2 = 0;
    char[] array_char = text.ToCharArray();
    for (int i = 0; i < array_char.Length; i++)
    {
      if (i % 2 == 0) sum1 += (array_char[i] - '0');
      else sum2 += (array_char[i] - '0') * 3;
    }
    int sum = sum1 + sum2;
    int K = sum;
    while (K % 10 != 0) K++;
    K = K - sum;
    return (char)(K + '0');
  }
```

Штрих-код EAN13 створюється за допомогою бібліотеки BarcodeLib.

Процедура створення штрих-коду відбувається наступним чином:

- до форми діалогового вікно CreateBarcode, передається інформація по приладу у вигляді масиву. Ця інформація зберігається в масиві Data, який ініціалізується в конструкторі форми;

- генерація штрих-коду відбувається за допомогою об’єкта класу Barcode в методі Encode, який приймає 13-ти значний код;
- при успішній генерації штрих-коду, метод Encode повертає графічний об’єкт класу Image, який відображається в елементі PictureBox;
- при необхідності графічний EAN13 код можна зберегти у форматі PNG.

Наведемо код функції графічної генерації штрих-коду:

```
private void button_Generate_Click(object sender, EventArgs e)
  {
    Barcode barcode = new Barcode();
    Color foreColor = Color.Black;
    Color backColor = Color.Transparent;
    string text = Convert(Data[0]);
    if (text == null)
    {
      MessageBox.Show("Неможливо отримати штрих код...");
      return;
    }
    Image img = barcode.Encode(TYPE.EAN13, text, foreColor, backColor,
    (int)(pictureBox1.Width * 0.8), (int)(pictureBox1.Height * 0.8));
    pictureBox1.Image = img;
        }
```

Для генрації коду необхідно натиснути Generate (див.рис.19), після відпрацювання методу button_Generate_Click у выкны выдобразиться новий штрих-код для обраного приладу.

Рисунок 19 – Вікно для створення штрих коду EAN13 (рисунок виконано самостійно)

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, яка гарантує, що система функціонує відповідно до вимог і очікувань користувачів.

Тестування програмного забезпечення – це процес оцінки та перевірки того, що програмний продукт або додаток виконує те, що повинен робити. Переваги тестування включають запобігання помилкам, зниження витрат на розробку та підвищення продуктивності [9].

Для перевірки надійності та якості розробленої системи були обрані такі типи тестування, як:

- функціональне тестування;
- тестування інтерфейсу.

Функціональне тестування було проведене для перевірки всіх функцій системи, таких як додавання, редагування, видалення і перегляд приладів.

Для здійснення функціонального тестування були обрані наступні сценарії, які необхідно перевірити:

- можливість зареєструватися в системі;
- можливість авторизуватися в системі;
- можливість нагадати пароль;
- можливість додати нового користувача в систему;
- можливість редагування даних користувача;
- можливість видалення користувача із системи;
- можливість редагування даних адміністратора;
- можливість додати прилад в БД;
- можливість видалити прилад із БД;
- можливість редагувати інформацію по приладу;
- можливість додати працівника КВП до БД;
- можливість видалити працівника КВП із БД;
- можливість редагувати інформацію по працівнику КВП;

- можливість додати об’єкт станції до БД;
- можливість видалити об’єкт станції із БД;
- можливість редагувати інформацію по об’єкту;
- можливість додати інформацію по станції до БД;
- можливість видалити інформацію по станції із БД;
- можливість редагувати інформацію по станції;
- можливість додати інформацію по відмові приладу до БД;
- можливість видалити інформацію по відмові приладу із БД;
- можливість редагувати інформацію по відмові приладу;
- можливість створювати SQL запити до БД для адміністратора системи;
- можливість пошуку приладу за певними критеріями;
- можливість створення штрих-коду EAN-13 для приладу;
- можливість переглядати певну інформацію;
- можливість створення відповідних звітів;
- можливість друку відповідних звіту;
- можливість надсилання повідомлень;
- можливість автоматичної розсилки повідомлень по приладам, які вимагають повторної перевірки, на електроні адреси станцій;
- можливість змінювати налаштування автоматичної розсилки повідомлень;
- можливість виходу в інтернет за допомогою вбудованого web-браузера;

Тестування можливості авторизування в системі та нагадування пароля відбувалося в такій послідовності:

- у діалоговому вікні входу в систему вводимо логін та пароль користувача, який зареєстрований в системі, та натискаємо кнопку «Увійти». Якщо логін та пароль правильний, то користувачу відкривається головне вікно програми. При цьому, якщо користувач входить в систему як адміністратор, то йому доступні пункти меню для роботи з БД;

- у діалоговому вікні входу в систему вводимо неправильний логін або пароль, та натискаємо кнопку «Увійти». При цьому з'являється відповідне повідомлення;
- перевіряємо можливість нагадування пароля. Натискаємо кнопку «Забули пароль», при цьому з'являється діалогове вікно в якому вводимо адресу електронної пошти користувача, який зареєстрований в системі та підтверджуємо її. Якщо електронна адреса зареєстрована в системі, то на цю адресу прийде лист з нагадуванням логіна та пароля. В противному випадку з'явиться відповідне повідомлення.

Тестування можливості додавання, редагування та видалення інформації здійснювалося за допомогою середовища SQL Server Management Studio, яке дозволяє взаємодіяти з БД. Також за допомогою SQL Server Management Studio було протестовано на працездатність відповідні SQL запити до БД, а також виконана їх оптимізація.

Після аналізу планів виконання запитів було прийнято рішення щодо створення індексів для зменшення часу виконання запиту. На рис. 20 наведено один зі створених індексів, який був створений для наведеного запиту.

```
SELECT type_device, surname, Item.name, name_station FROM WorkerKIP,
Device, Item, Station
WHERE WorkerKIP.id_worker=Device.id_worker AND
Device.id_item=Item.id_item
AND Item.id_station=Station.id_station AND type_device='НМШ2-900'

CREATE NONCLUSTERED INDEX index_typeDevice ON Device
(type_device) INCLUDE (id_worker, id_item)
```

Рисунок 20 – Запит на пошук інформації та створений індекс
(рисунок виконано самостійно)

На рисунку 21 представлена статистична інформація про виконання запиту, в тому числі затрачений час, та час ЦП.

Статистика виконання запиту до створення індексу

```
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

Час роботи SQL Server:
Час ЦП = 0 мс, витрачений час = 0 мс.
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

(зачеплено рядків: 18006)

(зачеплено один рядок)

Час роботи SQL Server:
Час ЦП = 93 мс, витрачений час = 570 мс.
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

Час роботи SQL Server:
Час ЦП = 0 мс, витрачений час = 0 мс.

Час виконання: 2024-07-02T22:46:00.4846202+03:00
```

Статистика виконання запиту після створення індексу

```
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

Час роботи SQL Server:
Час ЦП = 0 мс, витрачений час = 0 мс.
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

(зачеплено рядків: 18006)

(зачеплено один рядок)

Час роботи SQL Server:
Час ЦП = 15 мс, витрачений час = 339 мс.
Час синтаксичного аналізу та компіляції SQL Server:
час ЦП = 0 мс, час = 0 мс.

Час роботи SQL Server:
Час ЦП = 0 мс, витрачений час = 0 мс.

Час виконання: 2024-07-02T22:48:16.8523023+03:00
```

Рисунок 21 – Статистична інформація про виконання запиту (рисунок виконано самостійно)

Тестування можливість автоматичної розсилки повідомлень по приладам, які вимагають повторної перевірки, на електроні адреси станцій, відбувалося наступним чином:

- до БД додавалися прилади, у яких дата наступної перевірки менша за поточну дату (вимагають повторної перевірки);
- відповідні прилади повинні бути виділені червоним кольором в головній таблиці додатку;
- в файлі налаштувань авто-відправлення повідомлень MySettings.json (рис. 22), змінюючи властивості Date та Period можна примусово змусити систему розсилати повідомлення, щоб не чекати настання події відправлення;
- змінюємо властивості Date та Period таким чином, щоб виконувалося така умова: дата і час останньої розсилки повідомлень + періодичність розсилки повідомлень, менша за поточну дату і час;

- при цьому на відповідні електроні адреси станцій повинні прийти повідомлення з вкладеним звітом, в якому міститься інформація по приладам, які вимагають повторної перевірки.

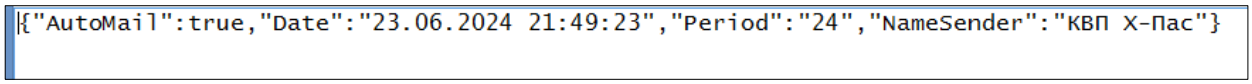


Рисунок 22 – Вміст файлу MySettings.json (рисунок виконано самостійно)

Налаштування авто-відправлення повідомлень зберігаються в JSON форматі з наступними властивостями:

- AutoMail – режим автоматичної розсилки повідомлень (true – включений, false - виключений);
- Date - дата і час останньої розсилки повідомлень;
- Period - періодичність автоматичної розсилки повідомлень (години);
- NameSender - ім’я відправника повідомлень.

Також важливим моментом тестування є перевірка працездатності елементів інтерфейсу користувача. Під час перевірки працездатності елементів інтерфейсу перевіряється:

- правильність відображення інформації у відповідних елементах;
- правильність роботи елементів керування (кнопки, прапорці, повзунки, пункти меню);
- правильність відображення інформації у взаємопов'язаних елементах (при виборі в ComboBox відповідної станції, в інший взаємопов'язаний ComboBox повинні підтягуватися об'єкти, які розташовані на обраній станції). Даний підхід реалізовано у вікні додавання та редагування інформації по приладу.

Також була проведена перевірка працездатності веб-застосунку для браузерів: Google Chrome (Version 120), Microsoft Edge (Version 119), Mozilla Firefox (Version 120), Opera (Version 105).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було застосовано практичні навички, здобуті в період навчання, для виконання кваліфікаційної роботи на тему «Розробка програмної системи, для підтримки обліку приладів залізничної автоматики».

Був проведений аналіз предметної області, з порівнянням функцій систем конкурентів, та сформовано вимоги до програмної системи.

Було спроектовано архітектуру програмної системи, структуру зберігання даних, створено UI/UX для системи обліку приладів залізничної автоматики, та розглянуто найцікавіші алгоритми та методи, які використовуються в програмній системі. В якості структури зберігання даних було обрано реляційну базу даних MS SQL.

Також було розглянуто питання пов’язані з UML проектуванням програмного забезпечення.

Було розроблено програмну систему, яка забезпечить надійну роботу приладів залізничної автоматики за рахунок постійного моніторингу стану приладів для своєчасного виявлення і усунення несправностей, а також допоможе значно підвищити безпеку та надійність залізничного транспорту.

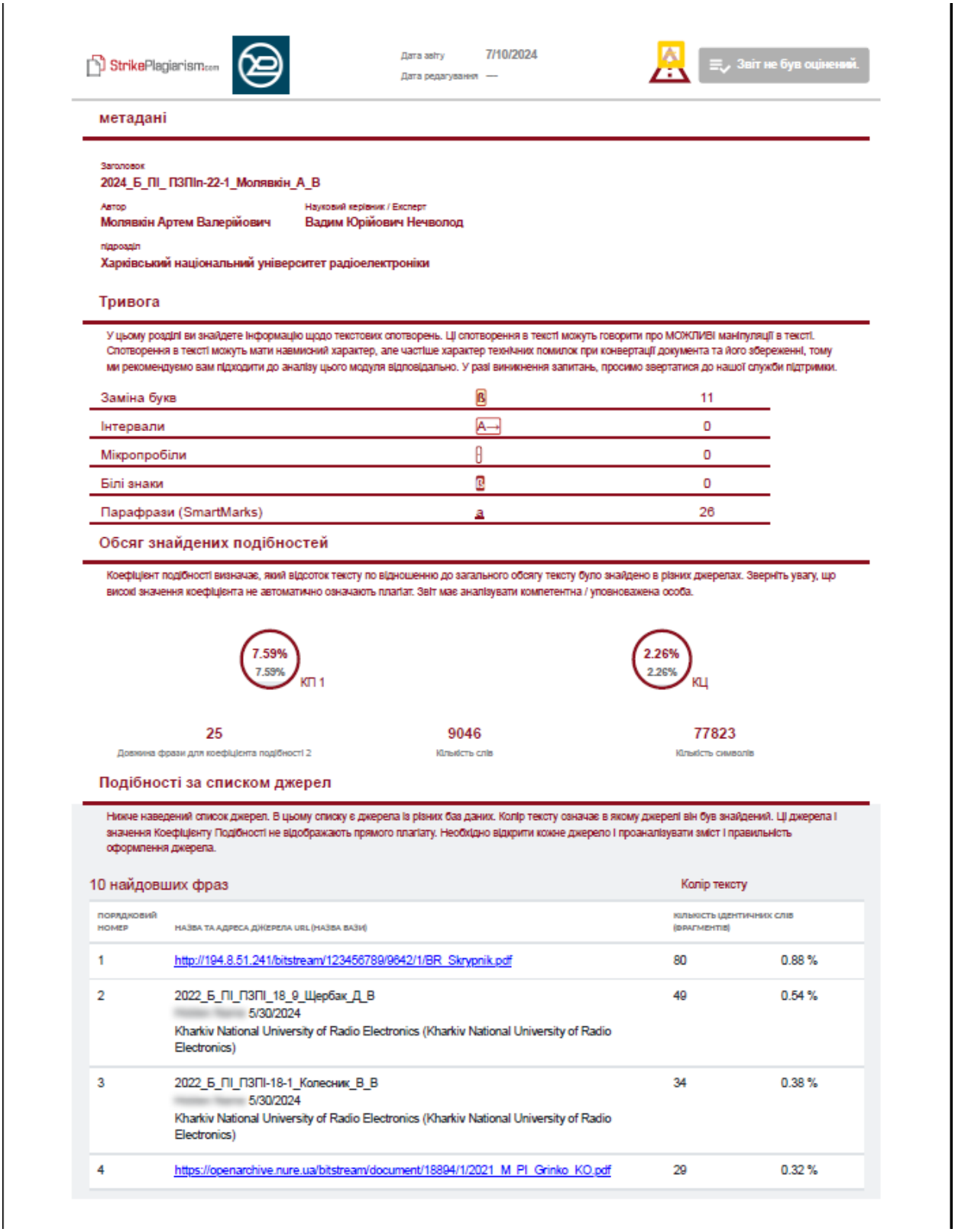
Для перевірки надійності та якості розробленої системи було проведене функціональне тестування та тестування інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Infor EAM. URL: https://fellowpro.com/products-and-services/digitale-transformation/infor-eam/ (дата звернення 16.05.2024)
2. IBM Maximo. URL: https://www.bpdzenith.com/products/maximo-application-suite/ (дата звернення 16.05.2024)
3. UpKeep. URL: https://upkeep.com/ (дата звернення 16.05.2024)
4. Fiix. URL: https://fiixsoftware.com/cmms/cmms-software/ (дата звернення 16.05.2024)
5. eMaint CMMS. URL: https://www.emaint.com/cmms/emaint-cmms-software/ (дата звернення 16.05.2024)
6. Ситнік Б.Т. Основи інформаційних систем і технологій. – Харків: Вид-во УкрДУЗТ, 2019. – 175с.
7. Martina Seidl. UML @ Classroom: An Introduction to Object-Oriented Modeling (Undergraduate Topics in Computer Science): Springer, 2022. – 458p.
8. Бублик В.В. Об'єктно-орієнтоване програмування: навч. посіб. – Київ: ІТ-книга, 2020. – 624с.
9. Kathi Kellenberger, Scott Shaw. Beginning T-SQL. – Berkeley: Apress, 2020. – 627p.
10. Anthony Molinaro. SQL Cookbook: O'Reilly Media, 2020. – 382p.
11. Desktop Guide (Windows Forms .NET). URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0 (дата звернення 16.05.2024)
12. ADO.NET Overview. URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview (дата звернення 18.05.2024)
13. Transact-SQL reference (Database Engine). URL: https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16 (дата звернення 18.05.2024)
14. Overview of ASP.NET Core MVC. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0 (дата звернення 28.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/10/2024
Дата редагування —

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп-22-1_Молявкін_А_В

Автор
Молявкін Артем Валерійович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

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

Заміна букв		11
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		26

Обсяг знайдених подібностей

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

7.59% 7.59% КП 1

2.26% 2.26% КЦ

25	9046	77823
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Подібності за списком джерел

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

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА ВАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://194.8.51.241/bitstream/123456789/9642/1/BR_Skrypnik.pdf	80	0.88 %
2	2022_Б_ПІ_ПЗПІ_18_9_Щербак_Д_В [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	49	0.54 %
3	2022_Б_ПІ_ПЗПІ-18-1_Колесник_В_В [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	34	0.38 %
4	https://openarchive.nure.ua/bitstream/document/18894/1/2021_M_PI_Grinko_KO.pdf	29	0.32 %

Рисунок А.1 – Перевірка на плагіат

ДОДАТОК Б

Слайди презентації

Рисунок Б.1-Слайд 1

Рисунок Б.2 - Слайд 2

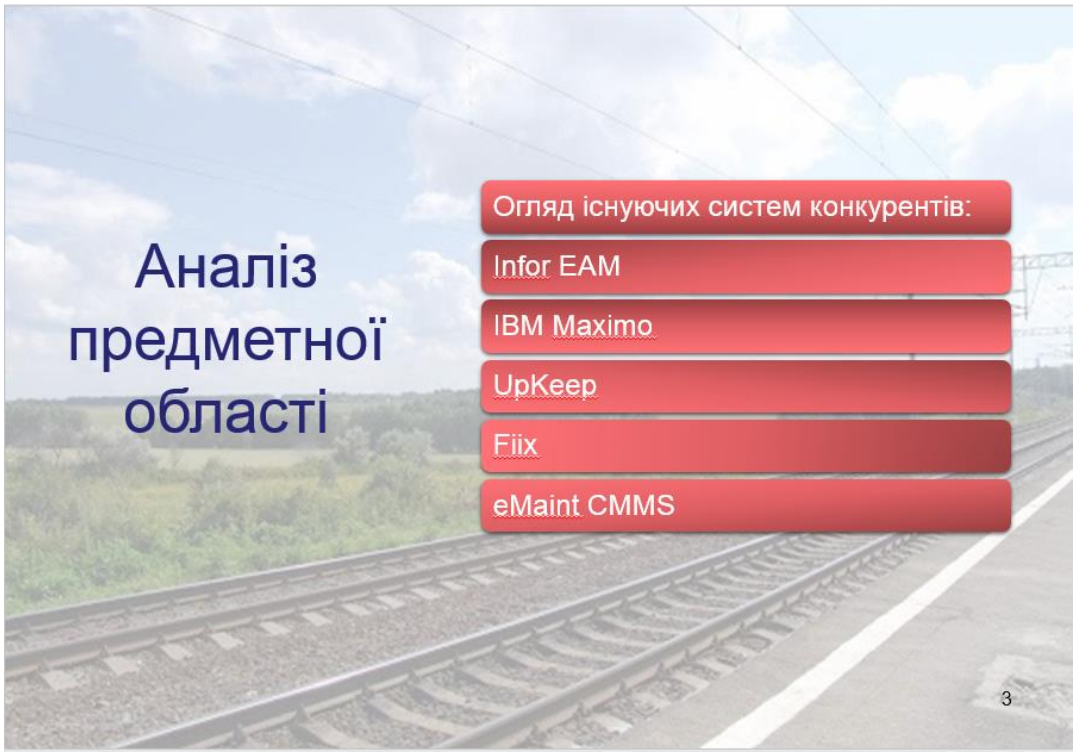


Рисунок Б.3 - Слайд 3

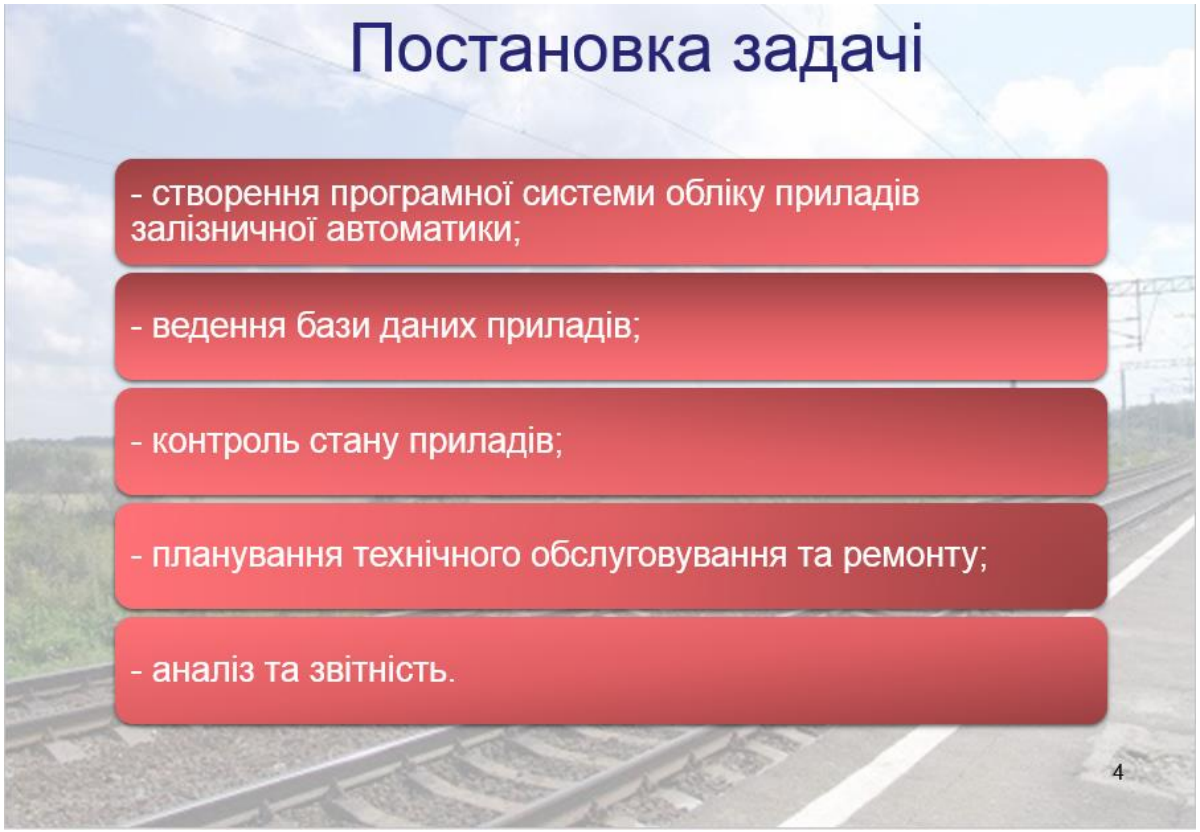


Рисунок Б.4 - Слайд 4

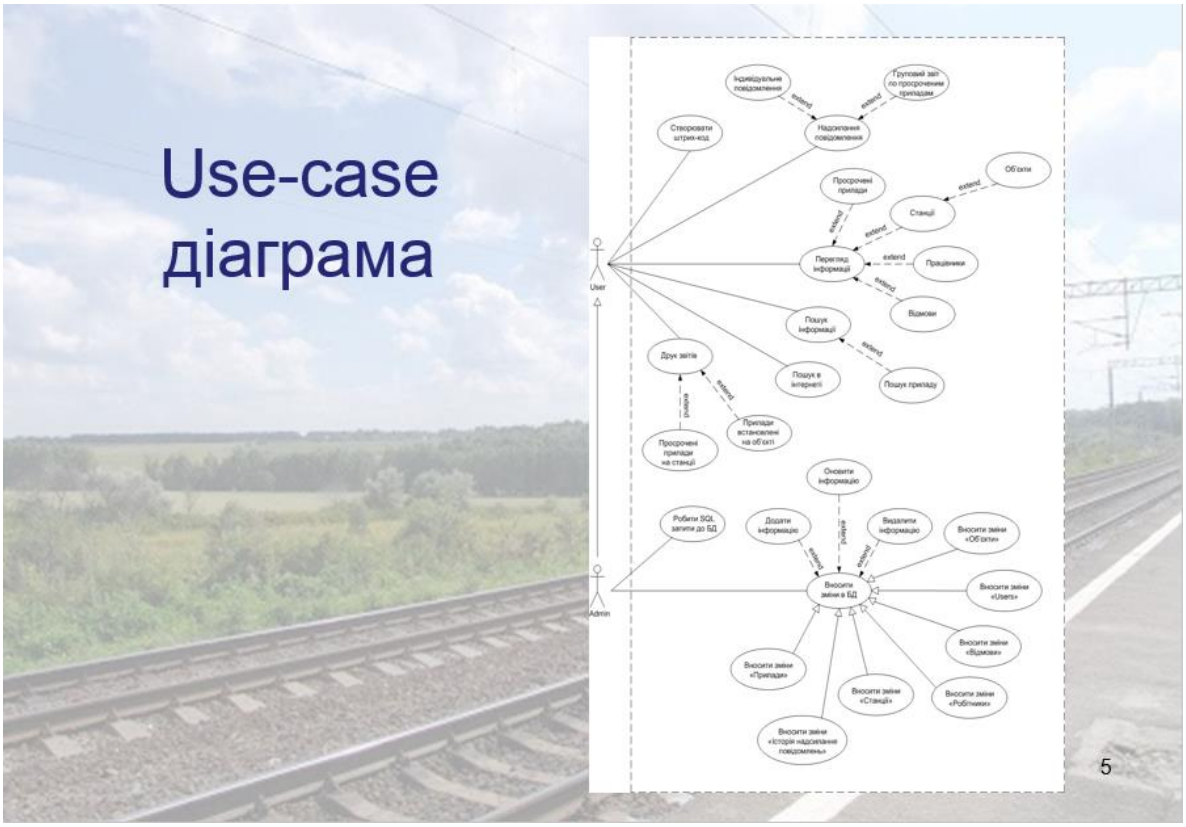


Рисунок Б.5 - Слайд 5

Рисунок Б.6 - Слайд 6

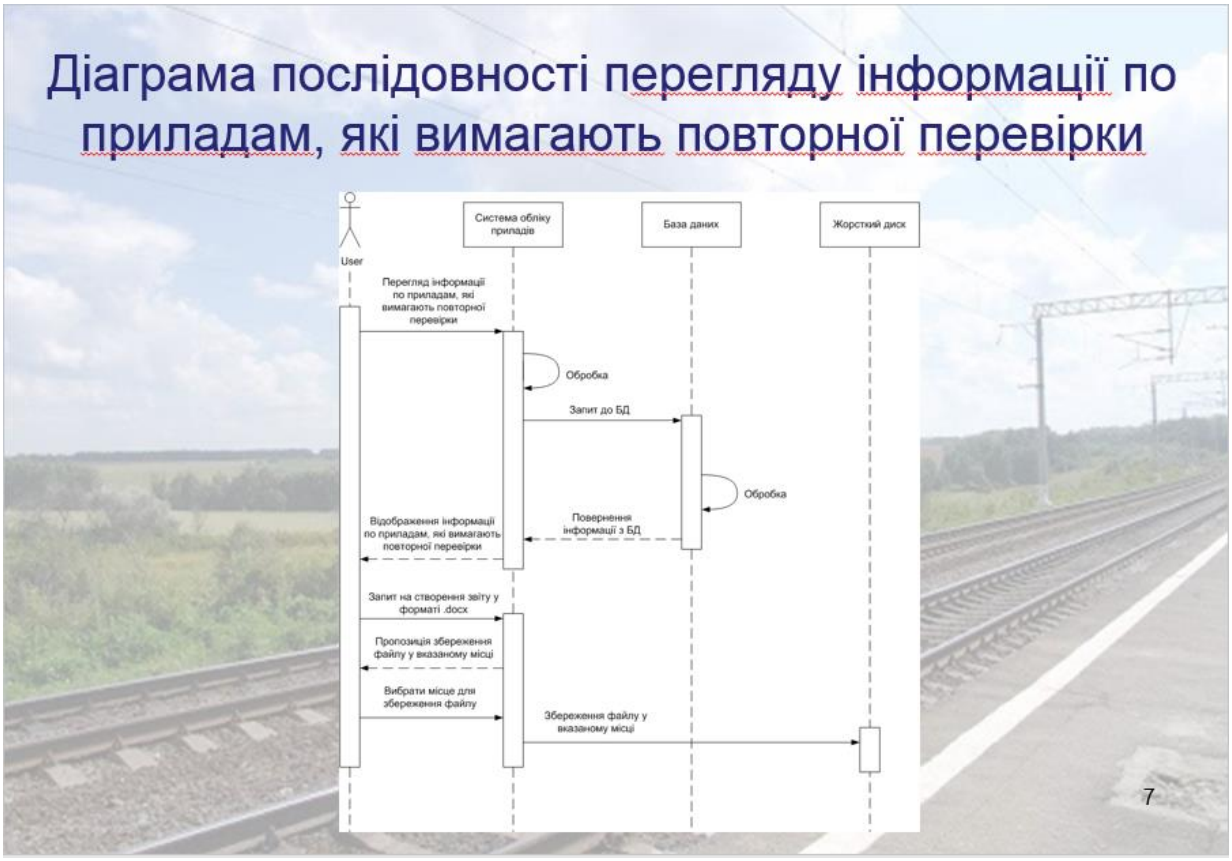


Рисунок Б.7 --Слайд 7

Рисунок Б.8 - Слайд 8

Рисунок Б.9 - Слайд 9

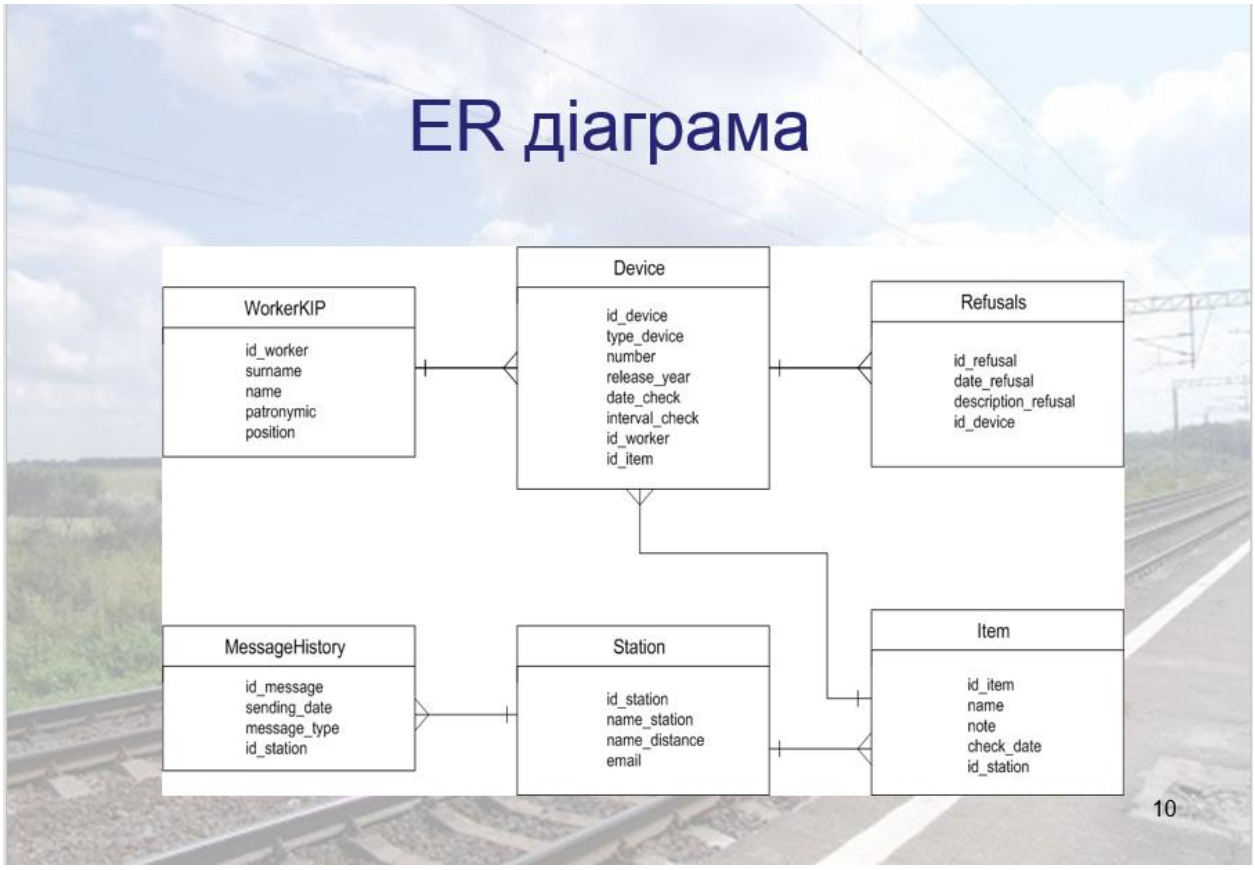


Рисунок Б.10 - Слайд 10

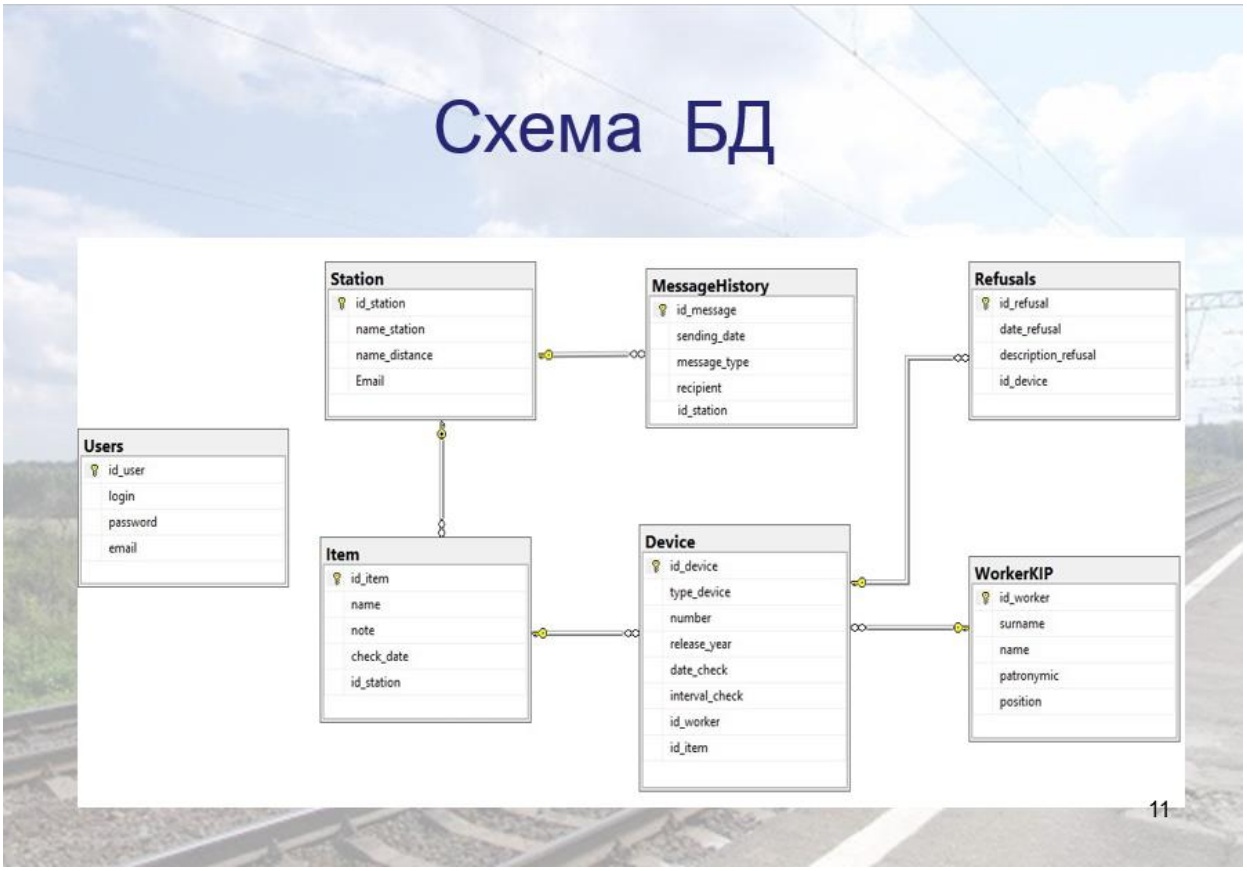


Рисунок Б.11 - Слайд 11

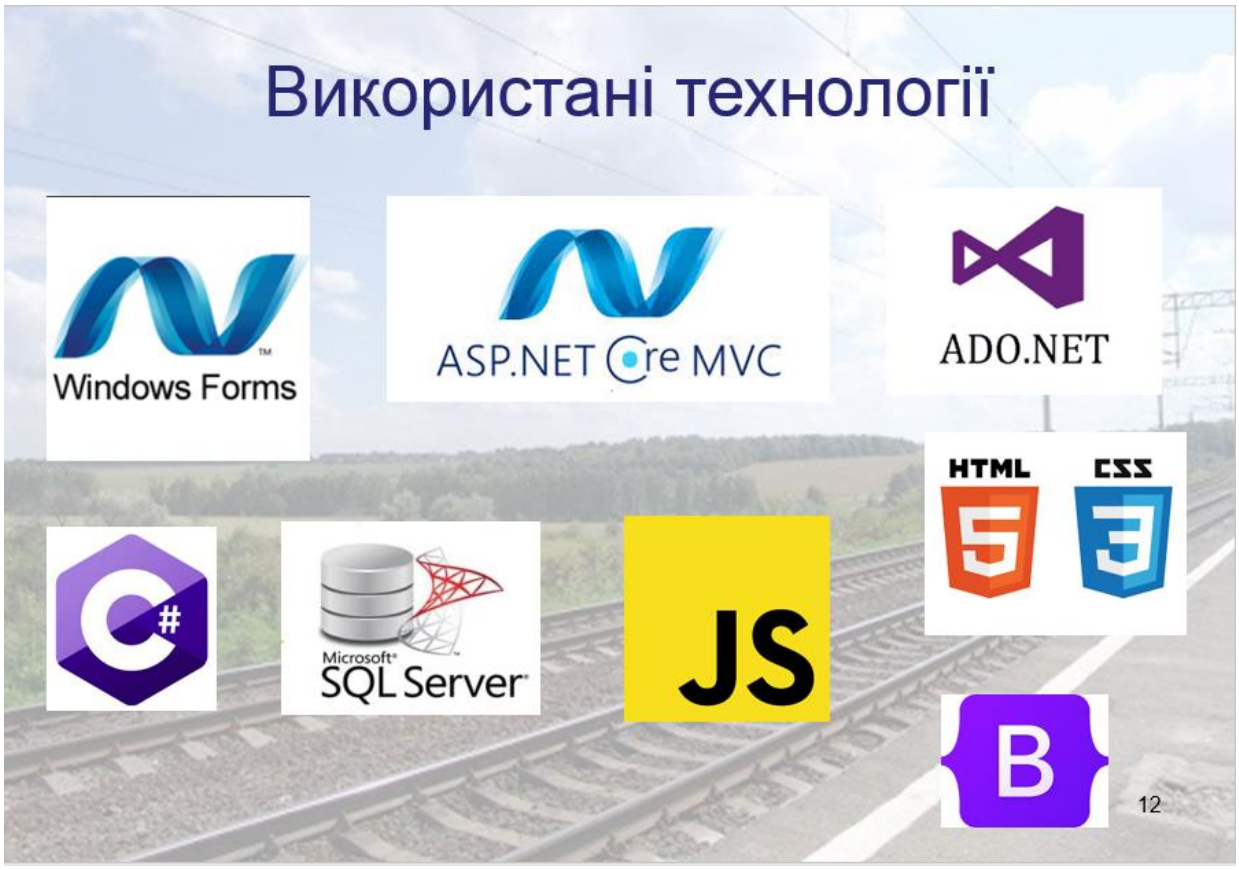


Рисунок Б.12 - Слайд 12

Рисунок Б.13 – Слайд 13

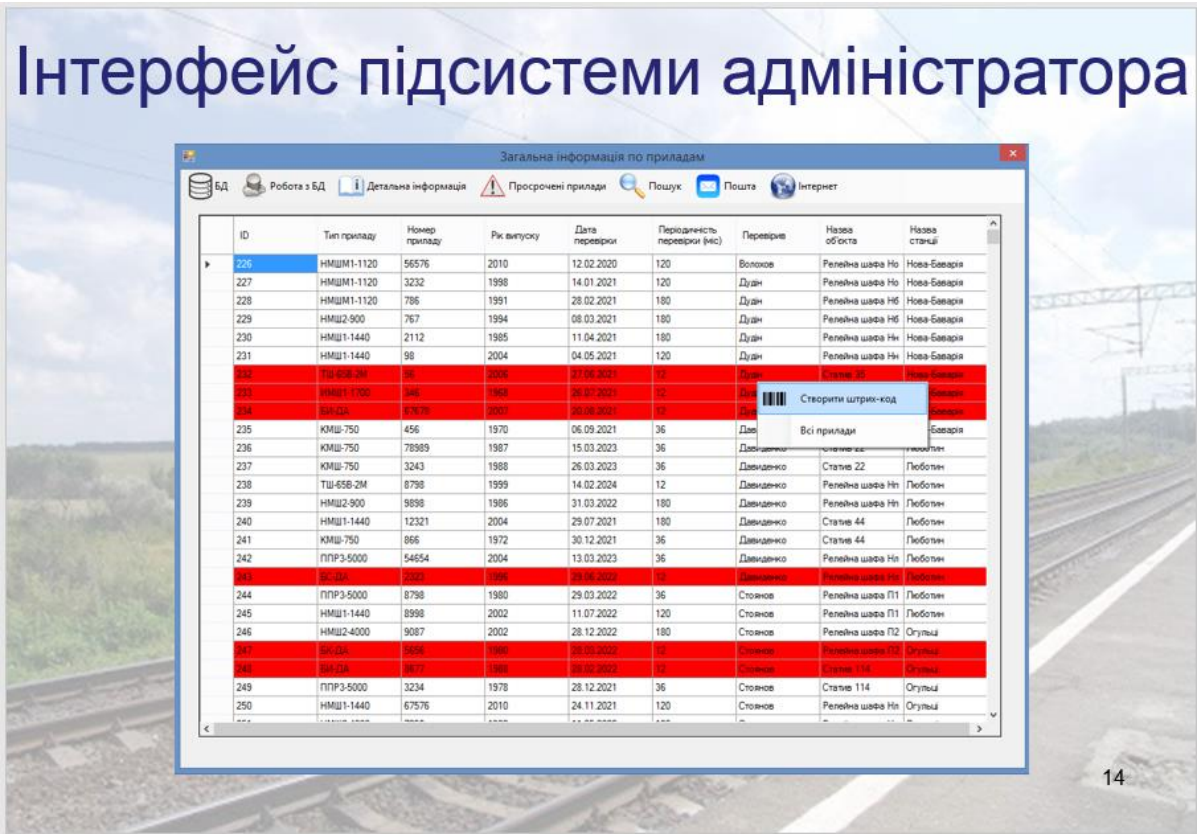


Рисунок Б.14 - Слайд 14

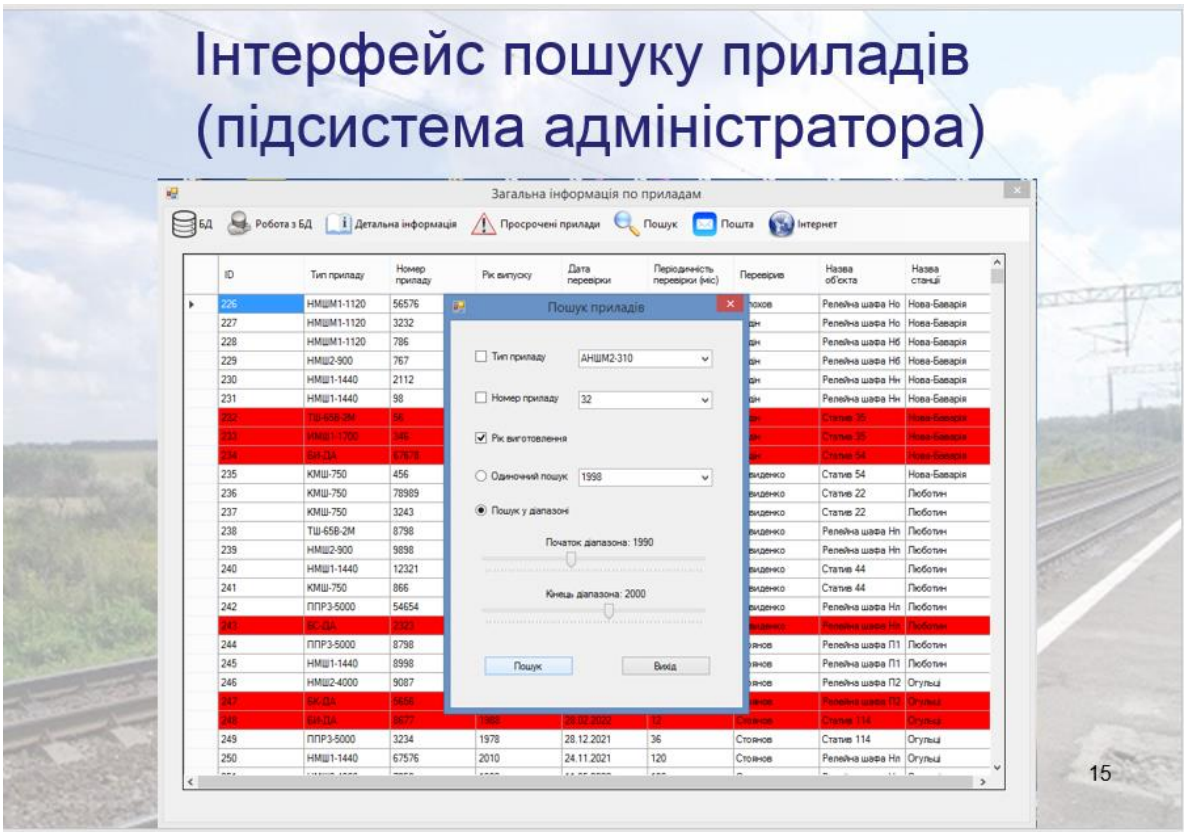


Рисунок Б.15 - Слайд 15

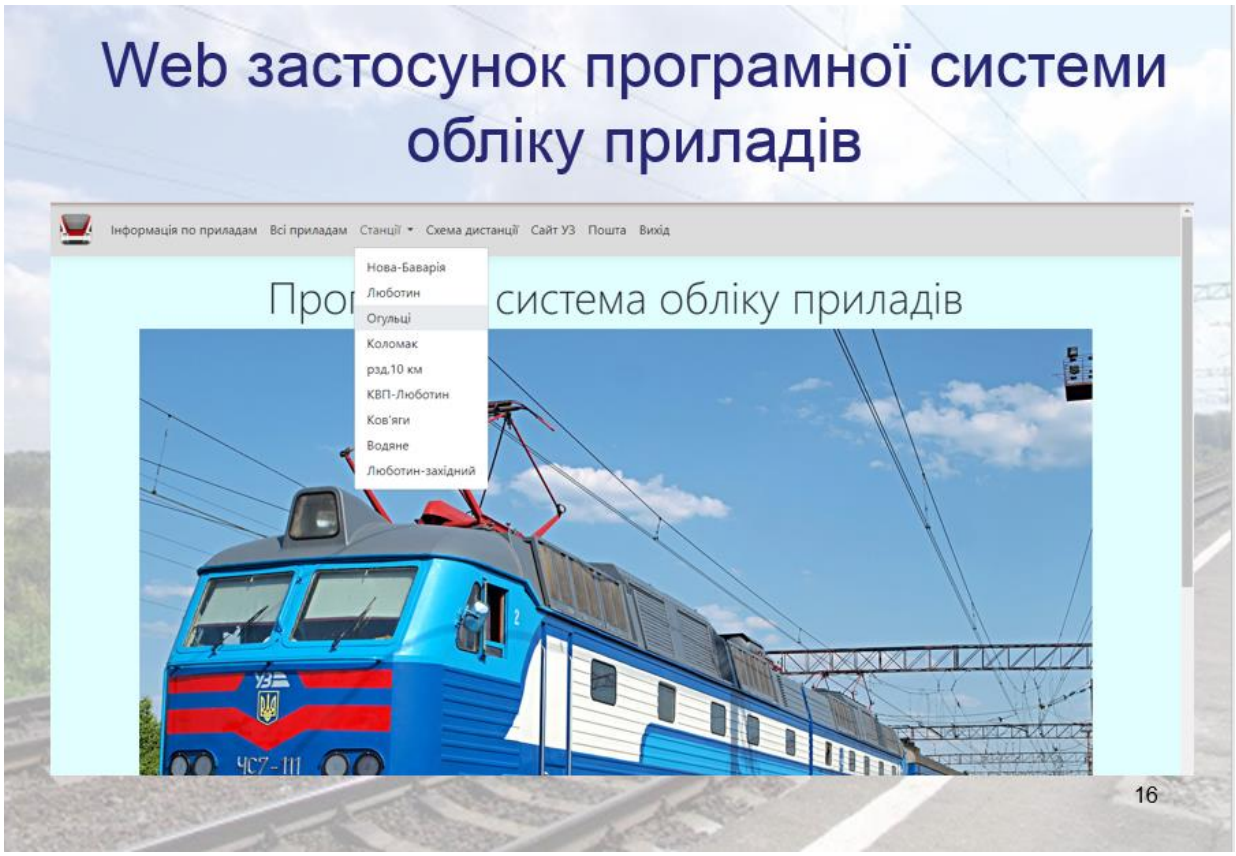


Рисунок Б.16 - Слайд 16

Рисунок Б.17 - Слайд 17

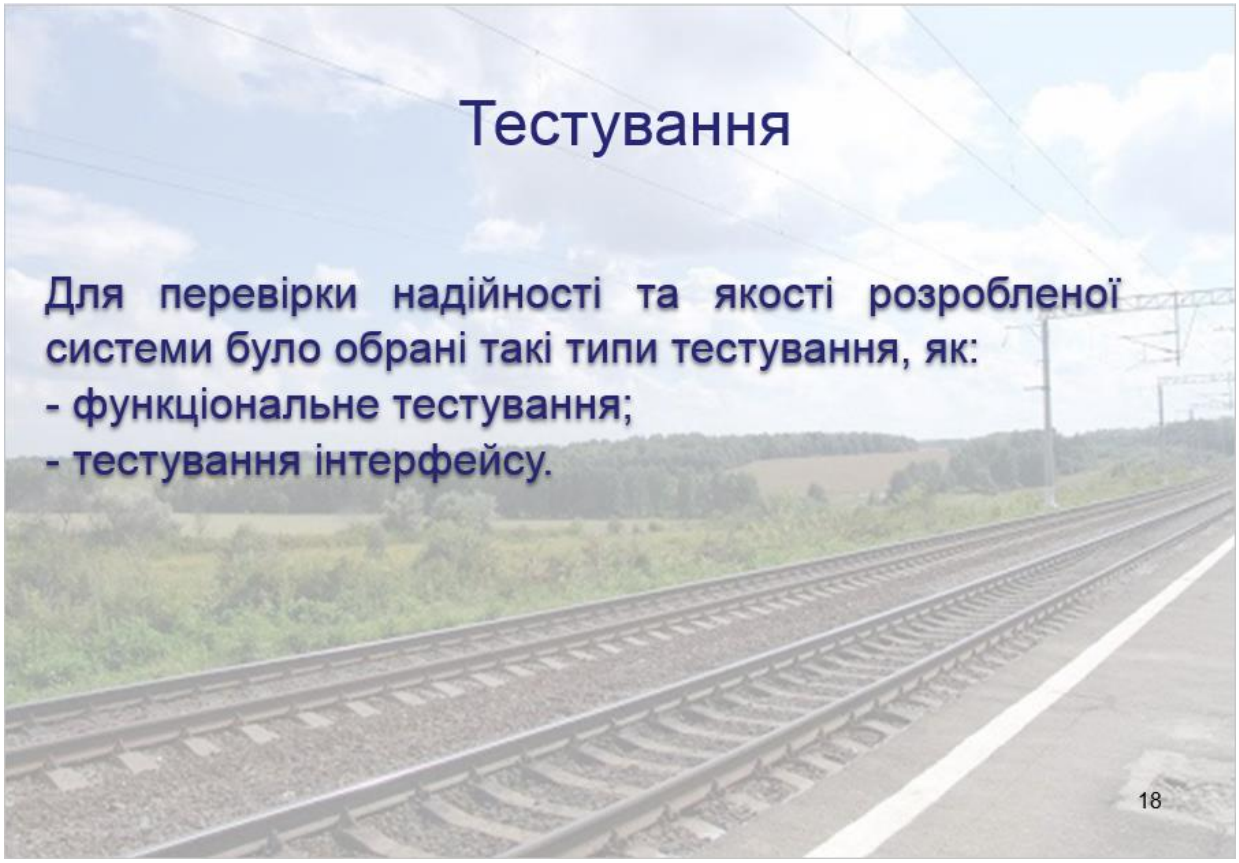


Рисунок Б.18 - Слайд 18

Рисунок Б.19 - Слайд 19

Рисунок Б.20 - Слайд 20

ДОДАТОК В

Приклад кодів програми

Файл Station.cs (клас створення і відправлення звітів на станції)

```
class Station
{
  private string ID;                    // ID станції
  private string Name;                  // назва станції
  private string Email;                 // email станції
  private string NameDistance;          // назва дистанції
  private DataTable dt;                 // просрочені прилади
  private static List<Station> sl = null;  // інформація по станціям
  private static SqlClass sc = new SqlClass();
  private static bool flag = false; // флаг заповнення списку
  private static string path = @"C:\Users\User\Мої файли\Temp\";
  private string PathSend = "";
  private int count = 0;
  private bool FileAvailability = false;

  public Station(string ID, string Name, string Email, string NameDistance)
  {
    this.ID = ID;
    this.Name = Name;
    this.Email = Email;
    this.NameDistance = NameDistance;
  }
  public void SetTable(DataTable dt) // функція заповнення таблиці з
  {                                  // простроченими приладами
    this.dt = dt;
    if (dt.Rows.Count > 0)
    {
      GetWordFile2();
      FileAvailability = true;
    }
    else FileAvailability = false;
  }
  private void GetWordFile2()  // функція створення звіту у форматі .docx
  {
    try
    {
      Application word = new Application();
      object missing = Missing.Value;
      Document doc = word.Documents.Add(ref missing, ref missing, ref
missing, ref missing);
      foreach (Section section in doc.Sections)
      {
        Range headerRange =
section.Headers[WdHeaderFooterIndex.wdHeaderFooterPrimary].Range;
        headerRange.Fields.Add(headerRange, WdFieldType.wdFieldPage);
        headerRange.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
        headerRange.Font.ColorIndex = WdColorIndex.wdBlue;
        headerRange.Font.Size = 14;
        headerRange.Text = "Дата створення документа: " + DateTime.Now;
      }
      foreach (Section wordSection in doc.Sections)
```

```
      {
        Range footerRange =
wordSection.Footers[WdHeaderFooterIndex.wdHeaderFooterPrimary].Range;
        footerRange.Font.ColorIndex = WdColorIndex.wdDarkRed;
        footerRange.Font.Size = 14;
        footerRange.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
        footerRange.Text = "Дистанція: " + NameDistance;
      }
      doc.Content.SetRange(0, 0);
      doc.Content.Text = "Станція: " + Name + Environment.NewLine;
      Paragraph para1 = doc.Content.Paragraphs.Add(ref missing);
      para1.Range.Text = "Прилади, які вимагають перевірки";
      para1.Range.InsertParagraphAfter();
      Table firstTable = doc.Tables.Add(para1.Range, dt.Rows.Count + 1,
dt.Columns.Count, ref missing, ref missing);
      firstTable.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;
      firstTable.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;
      for (int i = 0; i < dt.Columns.Count; i++)
      {
        firstTable.Cell(1, i + 1).Range.Text = dt.Columns[i].ColumnName;
      }
      for (int i = 0; i < dt.Rows.Count; i++)
      {
        for (int j = 0; j < dt.Columns.Count; j++)
        {
          if (j == 4)
          {
            string dateCheck = dt.Rows[i][j].ToString();
            dateCheck = dateCheck.Substring(0, dateCheck.IndexOf(' '));
            firstTable.Cell(i + 2, j + 1).Range.Text = dateCheck;
          }
          else
          {
            firstTable.Cell(i + 2, j + 1).Range.Text =
            dt.Rows[i][j].ToString();
          }
        }
      }
      count++;
      string pathFile = string.Format("{0}_({1}).docx", Name, count);
      PathSend = path + pathFile;
      doc.SaveAs2(PathSend);
      doc.Close(ref missing, ref missing, ref missing);
      word.Quit(ref missing, ref missing, ref missing);
    }
    catch (Exception ex) {}
  }
  public static void DataAnalysis(bool flag) //функція аналізу даних
  {
    SetListStation();
    IteratingStation(flag);
  }
  private static void SetListStation() //функція заповнення списку
  {
    if (flag) return;

    DataTable dt = sc.SqlRequest("SELECT id_station, name_station, email,
```

```
      name_distance FROM Station");
      sl = new List<Station>();
      flag = true;
      for (int i = 0; i < dt.Rows.Count; i++)
      {
        sl.Add(new Station(dt.Rows[i][0].ToString(),
        dt.Rows[i][1].ToString(), dt.Rows[i][2].ToString(),
        dt.Rows[i][3].ToString()));
      }
    }
    public static void DeleteFile() // функція видалення файлів
    {
      if (Directory.Exists(path))
      {
        string[] files = Directory.GetFiles(path);
        foreach (string s in files)
        {
          Thread.Sleep(100);
          try
          {
            File.Delete(s);
          }
          catch (IOException ex) {}
        }
      }
    }
    private static void IteratingStation(bool flag) // функція пошуку
    {                                               // просрочених приладів
      if (sl == null) return;
      foreach (Station st in sl)
      {
        string request = string.Format("Select Device.id_device AS ID,
        Device.type_device AS \"Тип приладу\", Device.number AS \"Номер
        приладу\", Device.release_year AS \"Рік випуску\", Device.date_check
        AS \"Дата перевірки\", Device.interval_check AS \"Періодичність
        перевірки (міс)\", WorkerKIP.surname AS \"Перевірив\", Item.name AS
        \"Назва об'єкта\" FROM WorkerKIP, Device, Item  WHERE
        WorkerKIP.id_worker = Device.id_worker AND Item.id_item =
        Device.id_item AND DATEDIFF(month, GETDATE(),  DATEADD(month,
        interval_check, date_check)) <= 1 AND Device.id_item IN (SELECT
        id_item FROM Item WHERE id_station = {0});", st.GetID());
        DataTable dt = sc.SqlRequest(request);
        st.SetTable(dt);
        st.SendMail(flag);
      }
    }
    private void SendMail(bool flag) // функція відправлення звіту
    {
      if (!FileAvailability) return;
      string text = "Прилади, які вимагають повтороної перевірки! " +
      DateTime.Now.ToString();
      if (flag) SqlMessageHistory.SqlInsert("Авто відправка звіту",
      this.Name);
      else SqlMessageHistory.SqlInsert("Ручна відправка звіту", this.Name);
      SendMailClass.SendMail(Email, "КВП Х-Пас", "Просрочені прилади!", text,
      PathSend);
    }
  }
```