МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних

Здобувача групи ІТ.м-22 Могили Юрія Олександровича
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ Юрій МОГИЛА
(підпис)

Керівник старший викладач кафедри ІТ, к.т.н., Ольга БОЙКО _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Могили Юрія Олександровича
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «11» грудня 2023 р.

3 Вхідні дані до кваліфікаційної роботи _____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) 1) Огляд технологій, що дозволять реалізувати поставлену задачу; 2) Постановка завдання й формування завдання дослідження; 3) Огляд технологій, що використовуються під час розробки додатків на Java, Spring; 4) Моделювання та проектування; 5) Програмна реалізація; 6) Аналіз результатів.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) _____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження предметної області		
2.	Аналіз програмних продуктів аналогів		
3.	Постановка задачі та формування завдань дослідження		
4.	Проектування інформаційної технології		
5.	Практична реалізація		
6.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Магістрант _____ Юрій МОГИЛА

Керівник роботи Старший викладач кафедри ІТ, к.т.н., Ольга БОЙКО

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 40 найменувань, додатків. Загальний обсяг роботи − 68 сторінок, у тому числі 35 сторінки основного тексту, 4 сторінки списку використаних джерел, 24 сторінок додатків.

Актуальність роботи полягає в вирішенні проблем автоматизації, безпеки та децентралізації в сфері інвентаризації контрольно вимірювальних приладів. Розробка програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних буде доцільною за рахунок попиту серед потенційних замовників.

Метою роботи є автоматизація процесу інвентаризації контрольно-вимірювальних приладів за рахунок розробки програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

Об’єктом роботи є облік та інвентаризація контрольно-вимірювальних приладів.

Ключові слова: програмний додаток, контрольно-вимірювальні прилади, JavaFX, Spring, хмарне зберігання даних.

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Огляд останніх досліджень і публікацій 8
1.2 Аналіз програмних продуктів-аналогів 9
1.3 Аналіз підходів в проектуванні систем 12
2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ РЕАЛІЗАЦІЇ 14
2.1 Постановка задачі та визначення мети 14
2.2 Вибір методів реалізації 15
2.2.1 Вибір мови програмування 15
2.2.2 Вибір технології розробки інтерфейсу 15
2.2.3 Вибір бази даних 16
2.2.4 Вибір архітектури 16
3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ 18
3.1 Моделювання варіантів використання 18
3.2 Структурно-функціональне моделювання 19
3.3 Діаграма розгортання 21
3.4 Моделювання даних 22
4 ПРАКТИЧНА ЧАСТИНА 24
4.1 Програмна реалізація серверної частини 24
4.2 Програмна реалізація системи сповіщень 28
4.3 Програмна реалізація десктопної частини 30
4.4 Реалізація функціоналу ведення звітності 36
4.5 Тестування розробленого функціоналу 38
ВИСНОВОК 40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41
ДОДАТОК А 45
ДОДАТОК Б 59

ВСТУП

Документообіг є невід'ємною складовою виробничого процесу в будь якому підприємстві. Поява комп'ютерів та інформаційних систем починає новий етап розвитку сфери ведення документообігу. Комп'ютерні технології виводять ефективність паперової роботи на новий рівень. Системи автоматизації виробничих процесів надають можливість контролю в реальному часі, а кількість часу на ведення паперової звітності скоротилася в десятки разів.

В усіх галузях пов'язаних с тривалим виробництвом є потреба в постійному контролі технологічних параметрів всіх процесів. Документальне підтвердження наявності та стану виробничих ресурсів називають інвентаризацією. Раніше це була складна та тривала паперова робота, адже помилка коштуватиме великих збитків компанії. Проте зараз більшість підприємств переходять на електронний документообіг, адже ефективність таких систем значно зменшує ризики в виробництві та пришвидшує всі процесі в компанії.

Вагомою перевагою онлайн сервісів над стаціонарними додатками є підтримка хмарного зберігання даних. За рахунок переваги в децентралізації все більше компаній обирають додатки з хмарними сервісами. Такий підхід мінімізує ризики втрати важливої інформації та збільшує гнучкість в підході до місця роботи, що є досить актуальним в часи популярності віддаленої роботи.

Після відмови від програмного забезпечення держави-агресора, в Україні виник попит на власні програмні засоби для ведення документообігу в компанії. В галузях промисловості є потреба в розробці системи ведення обліку-інвентаризації контрольно-вимірювальних приладів. Для вирішення нагальних потреб підприємств галузей промисловості, було прийнято рішення розробити програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

Об’єктом роботи є облік та інвентаризація контрольно-вимірювальних приладів. Предметом теми кваліфікаційної роботи магістра є облік та інвентаризація контрольно-вимірювальних приладів. Практична цінність виражається в підвищенні ефективності ведення обліку контрольно-вимірювальних приладів в підприємствах.

Отже, метою кваліфікаційної роботи магістра є розробка програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних, а також збір інформації про предметну область, планування робіт, аналіз ризиків та моделювання.

Для досягнення поставленої мети треба вирішити наступні задачі:

- проаналізувати існуючі технології розробки та обрати необхідні для реалізації проекту;
- виконати аналіз аналогів програмного додатку ведення обліку-інвентаризації контрольно-вимірювальних приладів;
- спроектувати програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних;
- реалізувати хмарну частину програмного додатку ведення обліку-інвентаризації контрольно-вимірювальних приладів;
- розробити інтерфейс та функціонал десктопної частини програмного додатку ведення обліку-інвентаризації КВП;
- протестувати функціонал розробленої системи програмного додатку ведення обліку-інвентаризації КВП.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Сучасні підприємства України широко використовують автоматизовані системи бухгалтерського обліку. В таких системах надаються широкі функціональні можливості з автоматизації процесу роботи підприємства [1, 2]. Використання таких додатків зменшує ризик виникнення помилки при внесенні та обробці даних, що зумовлена людським фактором. Проте розробники та підприємства виділяють мало уваги процесу інвентаризації на більш нижчих рівнях.

Інвентаризація як головний метод контролю ресурсів та майна в підприємстві є роботою не лише бухгалтерів [3, 4], а наприклад інженерів контрольно-вимірювальних приладів (КВП). В їх роботу входить постійний контроль справності КВП, які відповідають за роботу критичних виробничих вузлів підприємства.

Після виходу указу президента України №133/2017 від 28 квітня 2017 року, російські програмні засоби підпали під санкції, що в свою чергу ускладнило виконання роботи з інвентаризації КВП адже схожих до використовуваних раніше застосунків на українському ринку не представлено [2].

Процес виробництва складних технологічних рішень часто потребує децентралізації, адже неможливо виробити всі елементи в одному місці. Також умови повномасштабної війни децентралізація є одним з найефективніших методів уникнення ракетних обстрілів підприємства. Тому розроблюваний програмний додаток повинен надавати такі можливості за рахунок переміщення бази даних в хмарне середовище.

Враховуючи всі проаналізовані фактори, можна зробити висновок, що на українському ринку є сталий попит на розробку вузькоспеціалізованого програмного засобу для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

1.2 Аналіз програмних продуктів-аналогів

Серед програмних додатків для інвентаризації КВП, або як ще їх називають для ведення метрології [5], можна відзначити український додаток ІС-ПРО[6] та зарубіжний Metrology Asset Manager [7].

Додаток Metrology Asset Manager від компанії Hexagon надає можливості в реальному часі відстежувати продуктивність вимірювальних приладів. Віддалений моніторинг і сповіщення в режимі реального часу дозволяють легко вирішувати проблеми на ходу, запобігати простоям і скорочувати витрати на обслуговування обладнання.

Metrology Asset Manager розділено на чотири основні області з різними показниками: інформаційна панель активів, журнал подій, використання активів і загальна ефективність обладнання. На рисунку 1.1 зображено інтерфейс додатка Metrology Asset Manager.

Рисунок 1.1 – Інтерфейс додатка Metrology Asset Manager

Джерело: [7]

Серед українських аналогів популярним рішенням є програмний додаток ІС-ПРО розроблений українською компанією «Інтелект-Сервіс». Інформаційна система ІС-ПРО надає можливість ведення оперативного, бухгалтерського та управлінського обліків. Оперативний облік застосовує інформаційну базу для

прийняття управлінських рішень. Це надає можливість керувати запасами, закупками та збутом, фінансовими та договірними відносинами, а також здійснювати розрахунки з працівниками.

Незважаючи на можливість ведення обліку КВП цим додатком та підтримка хмарного зберігання даних, мінуси в роботі з даними задачами перевищують ефективність. Серед мінусів програмного додатку ІС-ПРО можна відзначити складність в опануванні користувачем, та потреба в більш потужному апаратному забезпеченні.

При визначенні критеріїв ефективного програмного додатку інвентаризації КВП необхідно опиратись на особливості клієнтів. Потенційні підприємства замовники зацікавлені в додатку який буде швидко працювати на застарілому апаратному забезпеченні, а інженери КПВ здебільшого мають низький рівень знань в роботі с програмними додатками тому інтерфейс має бути інтуїтивно зрозумілим та не ускладненим функціоналом. Також важливо щоб система підтримувала можливість оформлення звітів за відповідним українським стандартом ДСТУ, такий функціонал суттєво пришвидшить та облегшить роботу с ведення звітності інженера КВП.

Таблиця 1.1 – Порівняльний аналіз систем ІС-ПРО та Metrology Asset Manager.

Джерело: побудовано автором

Критерії	ІС-ПРО	Metrology Asset Manager
Висока швидкість роботи на старих системах	–	–
Простота інтерфейсу	–	–
Облік приладів і термінів повірки	–	+
Функціонал ведення звітності	+	–
Хмарне збереження даних	+	–
Підтримка документів Excel	+	–
Онлайн моніторинг навантаження КВП	–	+
Відповідність українському законодавству	+	–

Існує два основних підходи в збереженні даних при розробці програмних додатків – хмарне та локальне зберігання. Локальне зберігання даних це підхід в якому данні зберігаються фізично на системі разом з додатком. Такий підхід має значну перевагу в безпеці адже дані локально зберігаються на робочій машині. Популярність локального зберігання даних постійно зменшується, за рахунок розробки нових протоколів безпеки та збільшення популярності хмарних систем. Але повністю запит на локальне збереження даних не зникне, адже існують специфічні замовники які потребують повної безпеки даних або наприклад умови роботи пов’язані з проблемами в інтернет з’єднанні.

Хмарне збереження даних прибирає залежність від локального апаратного забезпечення адже данні зберігаються на віддаленому сервері. Такий підхід надає великі можливості в децентралізації робочого процесу. Також хмарне зберігання даних суттєво знижує ризик втрати даних, тому вихід з ладу робочої машини не вплине на цілісність даних. Хмарний підхід підтримують більшість популярних програмних рішень для ведення обліку та інвентаризації, що є однозначним доказом попиту та актуальності даного підходу.

Поєднання хмарного та локального зберігання даних буде гарним рішенням проблем кожного з підходів, система буде робити резервну копію даних з локального середовища та за потреби зможе підтримувати офлайн роботу з даними.

Проаналізувавши продукти аналоги представлені на ринку можна дійти висновку, що розроблюваний програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних буде користуватись попитом на українському ринку. Серед головних критеріїв ефективного рішення є: простота інтерфейсу, підтримка застарілого апаратного забезпечення та наявність функціоналу ведення обліку КВП з хмарним зберіганням бази даних.

1.3 Аналіз підходів в проектуванні систем

Перед етапом проектування програмного додатку, необхідно визначити архітектуру за якою буде розроблятися система. Серед популярних та доцільних архітектурних підходів є мікросервісна та клієнт-серверна архітектури.

Мікросервісна архітектура [8] базується на принципі розподілення великого програмного або веб-додатка на декілька незалежних сервісів які спілкуються між собою за допомогою певних правил реалізованих в API (application programming interface) [9]. Кожен сервіс таких додатків працює незалежно тому зміна чи додавання нового не вплине на працездатність інших модулів системи. Цей архітектурний стиль має перевагу в зручності розширення функціоналу, чим зарекомендував себе як гарне рішення для великих клієнт-орієнтованих проектах. Приклад схеми мікросервісної архітектури зображено на рисунку 1.2

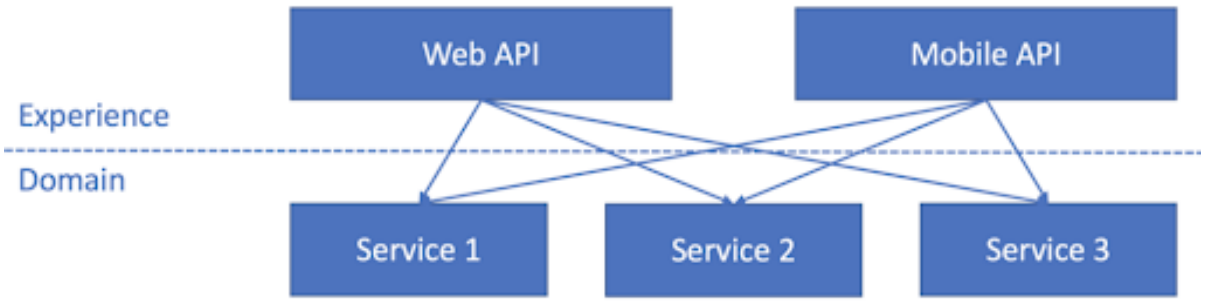

Рисунок 1.2 – Приклад мікросервісної архітектури

Джерело:[10]

Головним недоліком мікросервісної архітектури є її складність в порівнянні з більш простими архітекторами при розробці невеликих рішень не потребуючих її переваг в горизонтальному розширенні.

В свою чергу клієнт-серверна архітектура є більш широким поняттям принципи якого можна зустріти в мікросервісній також, тому необхідно уточнити, що в даному аналізі мається на увазі підхід с єдиним сервером який обмінюється даними з клієнтом, в нашому випадку програмним додатком, схема

клієнт-серверної архітектури наведено на рисунку 1.3. Такий підхід має суттєву перевагу в легкості та швидкості розробки невеликих за функціоналом рішень.

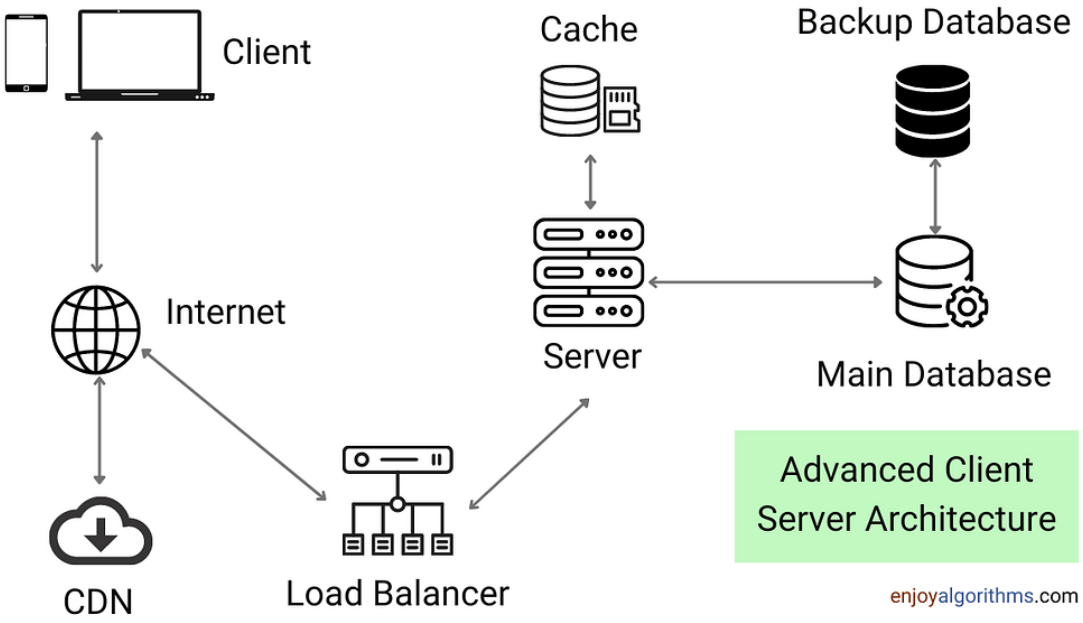

Рисунок 1.3 – Приклад клієнт-серверної архітектури

Джерело: [11]

В нашому випадку програмний додаток повинен зберігати бекап даних на віддаленому сервері, сервісів що будуть виконувати додаткові функції не передбачено, тому доречно буде використати саме клієнт серверний архітектурний підхід для розробки програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ РЕАЛІЗАЦІЇ

2.1 Постановка задачі та визначення мети

Метою роботи є автоматизація процесу інвентаризації контрольно-вимірювальних приладів за рахунок розробки програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних. Об’єктом кваліфікаційної роботи магістра є облік та інвентаризація контрольно-вимірювальних приладів, предметом роботи є облік та інвентаризація контрольно-вимірювальних приладів.

Розроблений програмний додаток повинен надавати наступні можливості:

- автоматизувати процес інвентаризації з використанням бази даних;
- зчитувати та записувати данні в форматі Microsoft Excel;
- можливість автоматичного збереження даних в хмарній базі даних;
- функціонал оформлення звітності в форматі PDF;
- можливість ведення журналу повірок та ремонтів;
- функціонал сповіщення користувача про потребу в повірці.

Для досягнення поставленої мети необхідно провести аналіз предметної області, обрати засоби та методи вирішення задачі, виконати проектування системи, розробити хмарну та десктопну частину програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних, а також інтегрувати систему сповіщень користувача та протестувати розроблений функціонал.

Цільовою аудиторією використання розроблюваного програмного продукту є підприємства які мають потребу в інвентаризації контрольно-вимірювальних приладів, а саме інженери контрольно-вимірювальних приладів.

Розробка призначена для підвищення якості процесу інвентаризації контрольно-вимірювальних приладів за рахунок інтеграції інформаційної системи та хмарних технологій. Планування робіт дипломної роботи представлено в додатку А.

2.2 Вибір методів реалізації

2.2.1 Вибір мови програмування

Обираючи технології реалізації необхідно опиратись на особливості використання та апаратне забезпечення замовника. Існує багато мов підтримуючих розробку десктопних додатків серед яких є: Python, C++, Java, JS [12].

Необхідно одразу виключити мови програмування які погано працюють зі старими системами, адже апаратне забезпечення цільової аудиторії, тобто підприємств, залишається досить застарілим. Тому серед потенційно прийнятних технологій залишаються C++ та Java.

Другим критерієм вибору є сумісність додатку з різними операційними системами, адже можливість децентралізованої роботи з додатком передбачає також можливу роботу на іншій операційній системі наприклад Linux. Таким критеріям відповідає мова програмування Java. Вона однаково працює за рахунок Java Virtual Machine(JVM) на усіх операційних системах. Також для мови програмування Java розроблено багато бібліотек та фреймворків як для роботи с базами даних так и для розробки додатків с візуальним інтерфейсом – JavaFX.

2.2.2 Вибір технології розробки інтерфейсу

Найпоширенішою платформою для розробки GUI(Graphical user interface) – застосунків на Java є JavaFX [13, 14]. Цей інструментарій дозволяє створювати додатки з якісною графікою завдяки використанню апаратного прискорення графіки і можливостей графічного процесора. Основною характеристикою розробки на JavaFX є використання декларативного опису інтерфейсу з використанням мови розмітки FXML, а також стилізація з використанням CSS. JavaFX підтримує різні способи компонування графічного інтерфейсу найпоширеніший з яких є JavaFX Scene Builder [13]. Основними перевагами

JavaFX Scene Builder є зручний інтерфейс для створення макета, проста інтеграція з Java IDE та автоматичне генерування коду FXML.

2.2.3 Вибір бази даних

Для роботи з базами даних, в Java є набір інтерфейсів і класів під назвою JDBC [15]. Програма спілкується с базою даних за рахунок набору інструкцій пад назвою JDBC-driver [16], які реалізовані окремо для кожної бази даних наприклад: MySQL, Oracle чи PostgreSQL.

Базою даних для програмного додатку було обрано PostgreSQL [17], який має низку переваг над MySQL, за рахунок продуктивності, безпеки та зручності у розгортанні з використанням Docker [18, 19].

2.2.4 Вибір архітектури

Програмний додаток буде побудовано за архітектурою Model-View-Controller (MVC) [20]. Обравши підхід MVC при розробці десктопного додатку з візуальним інтерфейсом на основі JavaFX, я спирався на його здатність ефективно розділити логіку додатку на три основні компоненти. Модель відповідає за управління даними та бізнес-логікою, що дозволяє відокремити логіку обробки даних від представлення та контролю. Представлення відповідає за візуальне відображення інформації, а контролер виконує обробку подій та взаємодіє як з моделлю, так і з представленням [20].

Цей підхід сприяє вищій модульності та підтримує легку модифікацію і розширення коду. Модель може бути змінена без впливу на візуальний інтерфейс, і навпаки. Також цей підхід полегшує тестування, надаючи можливість проводити автономні тести для кожного компонента. Крім того, MVC забезпечує вищу читабельність та організацію коду, що є ключовими аспектами при роботі над складними проектами. Узгоджуючи взаємодію між компонентами, підхід MVC сприяє створенню добре структурованих та ефективних десктопних додатків на платформі Java.

Для реалізації хмарного зберігання даних було обрано клієнт-серверну архітектуру додатку. Тобто буде розроблено два окремих програмних модулі: десктопний програмний додаток який буде підтримувати базовий функціонал системи обліку без підключення до мережі інтернет, а також хмарний API додаток який буде розширювати функціонал десктопного додатку надаючи можливість авторизуватися в обліковий запис користувача, сирнхронізуватися з хмарною базою даних та налаштувати функціонал сповіщення. Для розробки хмарної частини було обрано Spring framework в основі архітектури якого лежить вже знайомий нам архітектуриний шаблон MVC [21], але враховуючи що розроблятиметься саме серверна частина за «View» тобто представлення буде відповідати десктопна частина встановлена на стороні клієнта.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Моделювання варіантів використання

Перед початком практичної реалізації програмного додатку необхідно провести проектування інформаційної системи для уникнення помилок та розбіжності в розумінні кінцевого продукту с замовником.

Для визначення функціональних можливостей розроблюваного продукту необхідно провести моделювання системи з використанням діаграми варіантів використання (Use Case Diagram) [22]. Діаграма варіантів використання є однією з найпростіших поведінкових діаграм, що використовується в нотації Unified Modeling Language (UML) [23]. Вона використовується для опису функціональних вимог до програми або системи, також вона є дуже зрозумілою навіть для замовників не знайомих з IT сферою, чим допомагає уникнути непорозумінь на стадії проектування.

Діаграма варіантів використання в нотації UML представлена на рисунку 3.1.

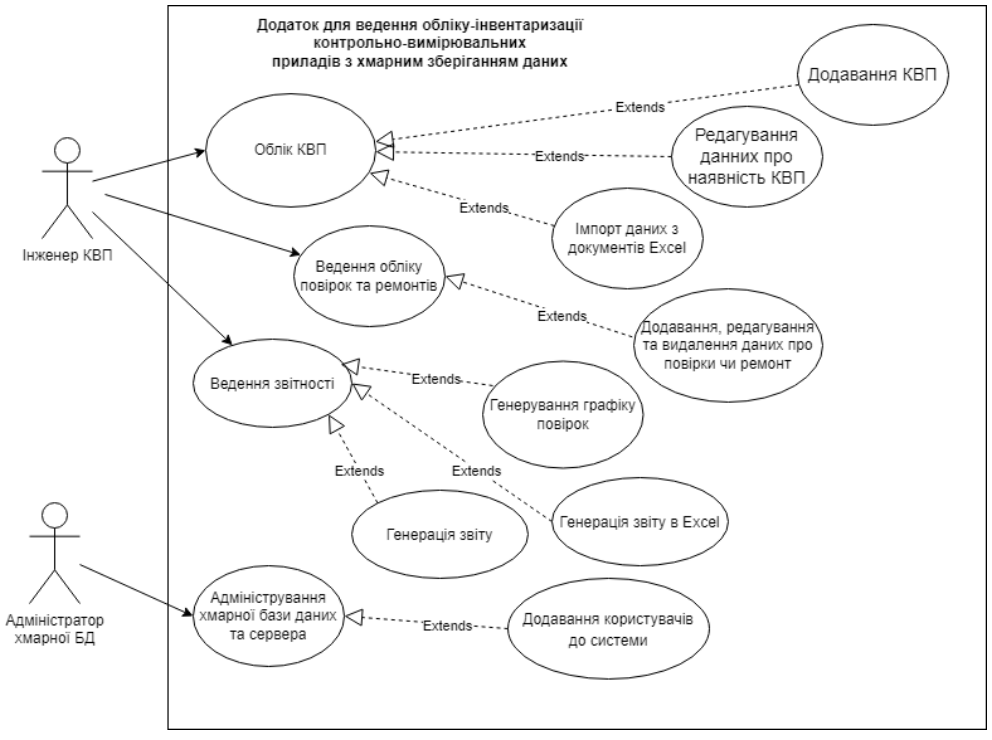

Рисунок 3.1 – Діаграма варіантів використання

Джерело: побудовано автором

3.2 Структурно-функціональне моделювання

Для повного відображення структури, функцій та потоків даних системи необхідно провести структурно-функціональне моделювання з використанням нотації IDEF0 [24, 25].

Функціональне моделювання програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних в нотації IDEF0 представлено на рисунку 3.2.

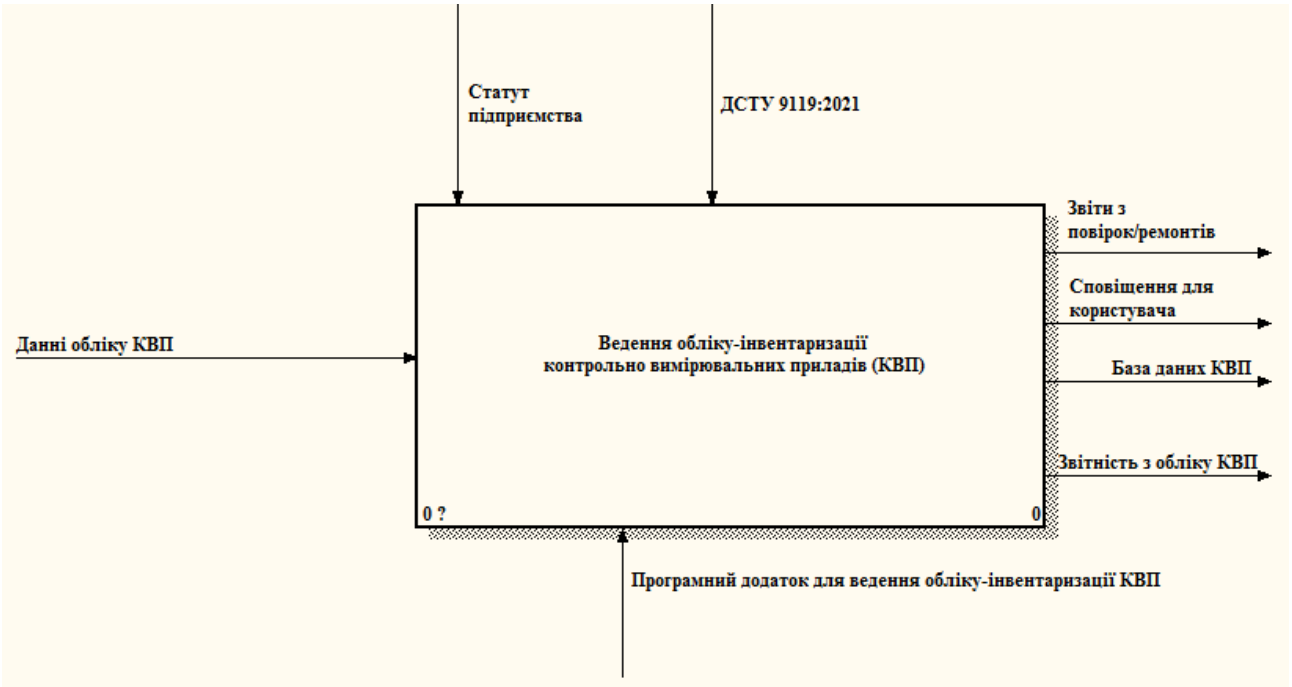

Рисунок 3.2 – Функціональна діаграма IDEF0

Джерело: побудовано автором

Вхідними даними системи є дані обліку КВП, а результатом діяльності програмного додатку є база даних КВП та звіти. Механізмом керування виступає ДСТУ 9119:2021 Метрологія [26], а механізмом виконання роботи є сам програмний додаток ведення обліку-інвентаризації КВП.

Декомпозиція функціональної моделі програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних представлено на рисунку 3.3.

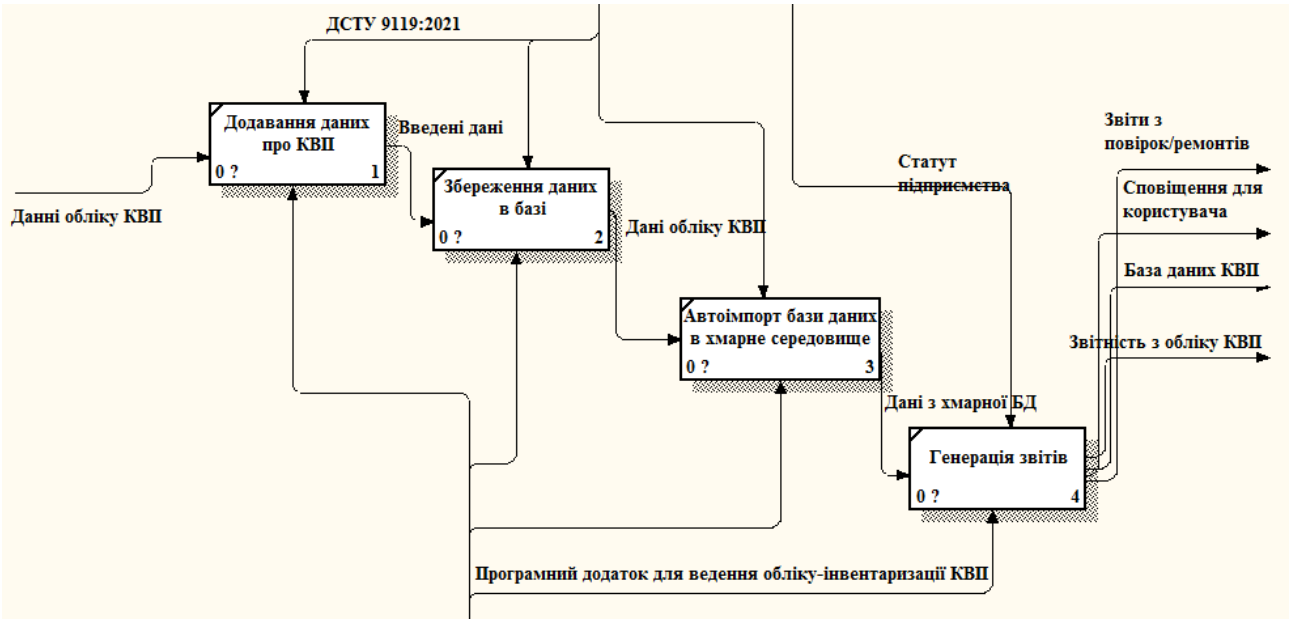

Рисунок 3.3 – Декомпозиція функціональної моделі

Джерело: побудовано автором

Декомпозиція функціональної моделі системи відображає всі механізми задіяні в процесі досягнення мети. Такий підхід допомагає більш детально відобразити аспекти роботи функцій та потоків даних в проекті.

3.3 Діаграма розгортання

Ще однією корисною діаграмою з нотації UML є діаграма розгортання [27]. Вона відображає обчислювальні вузли, компоненти та об’єкти, що виконуються на цих вузлах.

На рисунку 3.4 представлена діаграма розгортання програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних

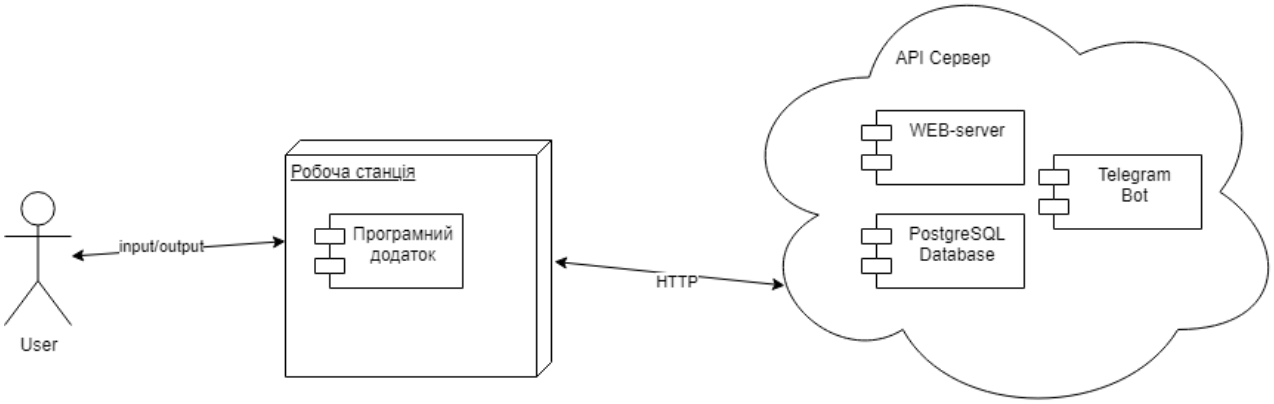

Рисунок 3.4 – Діаграма розгортання

Джерело: побудовано автором

В хмарному середовищі встановлено API-сервер базу даних та Telegram бота, в свою чергу програмний додаток який встановлено на робочу станцію користувача обмінюється даними з хмарним середовищем з допомогою HTTP сесії .

3.4 Моделювання даних

При побудові моделі бази даних програмного додатку ведення обліку КВП, необхідно визначити перелік полів які будуть зберігатись в процесі інвентаризації. Основними сутностями є: номер виробу, назва, реєстраційний номер, дата останньої повірки, період повірки, та нотатки. Для зберігання даних про користувача додатково необхідно створити таблицю бля зберігання облікових записів.

Логічна модель бази даних зображена на рисунку 3.5.

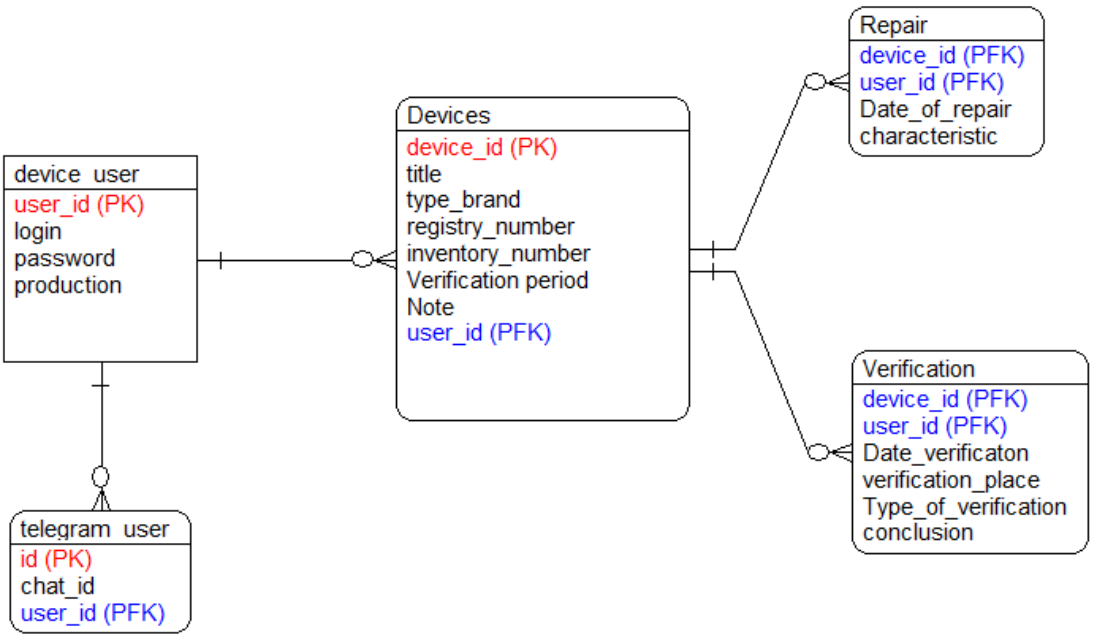

Рисунок 3.5 – Логічна модель бази даних

Джерело: побудовано автором

Діаграма потоків даних (DFD) [28, 29] допомагає візуально відобразити переміщення даних в системі ведення обліку КВП.

На рисунку 3.6 представлена діаграма потоків даних.

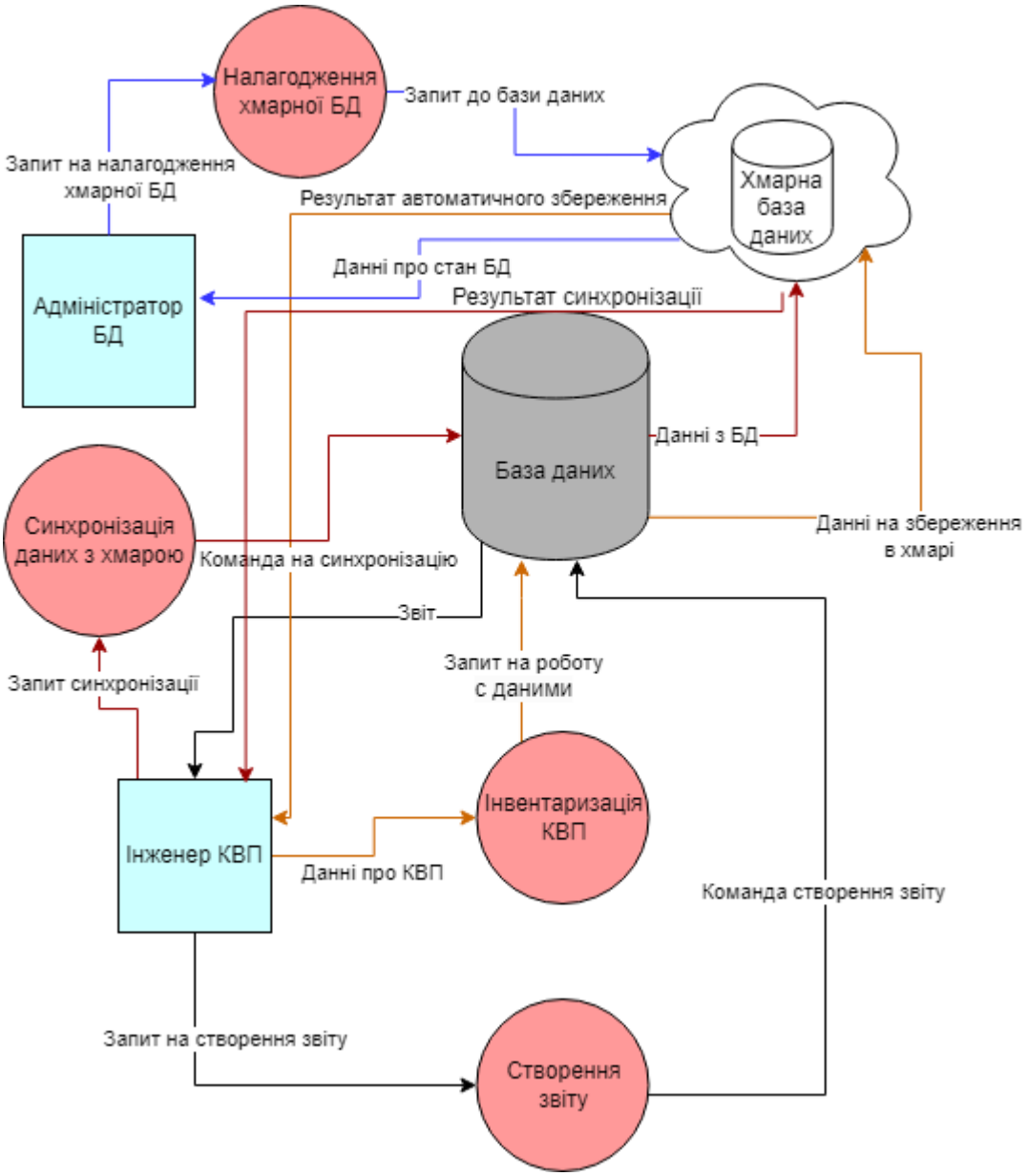

Рисунок 3.6 – Діаграма потоків даних

Джерело: побудовано автором

Як можемо побачити процес синхронізації з хмарною базою даних відбувається автоматично при роботі с даними або вручну за потреби користувача.

4 ПРАКТИЧНА ЧАСТИНА

4.1 Програмна реалізація серверної частини

Хмарна частина додатку була реалізована з використанням фреймворку Spring Boot [30, 31]. Він є потужним інструментом для розробки веб-орієнтованих додатків, до його складу входить інструмент Spring Security який гарантує безпеку авторизації. Налаштування безпеки авторизації реалізовано в класі «WebSecurityConfig» який зображено на рисунку 4.1.

```
@Configuration
public class WebSecurityConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)
                .formLogin(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(authorize →
                        authorize.requestMatchers( ...patterns: "/api/login").permitAll()
                                .anyRequest().authenticated()
                )
                .requestCache(cache → cache.requestCache(new NullRequestCache()))
                .exceptionHandling(handling → handling.authenticationEntryPoint(new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED)

        return http.build();
    }
    @Bean
    public UserDetailsManager users(DataSource dataSource) { return new JdbcUserDetailsManager(dataSource); }

    @Bean
    public AuthenticationManager authenticationProvider(PasswordEncoder passwordEncoder,
                                                        DatabaseUserDetailsService userDetailsService) {
```

Рисунок 4.1 – налаштування безпеки авторизації

Джерело: зроблено автором

Обмін даними між клієнтом та сервером розроблено з використанням HttpSession [32]. Для реалізації автоматичного входу в раніше авторизований обліковий запис додаток зберігає сесію в окремому файлі. Контролер авторизації зображено на рисунку 4.2.

```
@RestController
@AllArgsConstructor
public class AuthController {
    1 usage
    private final AuthenticationManager authenticationManager;
    1 usage
    private final SecurityContextRepository contextRepository;

    @PostMapping("/api/login")
    public void login(@RequestBody LoginDto loginDto, HttpServletRequest request, HttpServletResponse response) {
        final UsernamePasswordAuthenticationToken token =
                new UsernamePasswordAuthenticationToken(loginDto.getLogin(), loginDto.getPassword());
        final Authentication authenticated = authenticationManager.authenticate(token);

        contextRepository.saveContext(SecurityContextUtils.forAuthentication(authenticated), request, response);
    }
}
```

Рисунок 4.2 – Контролер авторизації користувача

Джерело: зроблено автором

Для авторизації до облікового запису користувачу потрібно натиснути на кнопку «Увійти до облікового запису» та ввести свій логін та пароль, після успішної авторизації користувач може завантажити базу даних з серверу та зберегти на ньому. Приклад успішної авторизації до облікового запису наведено на рисунку 4.3.

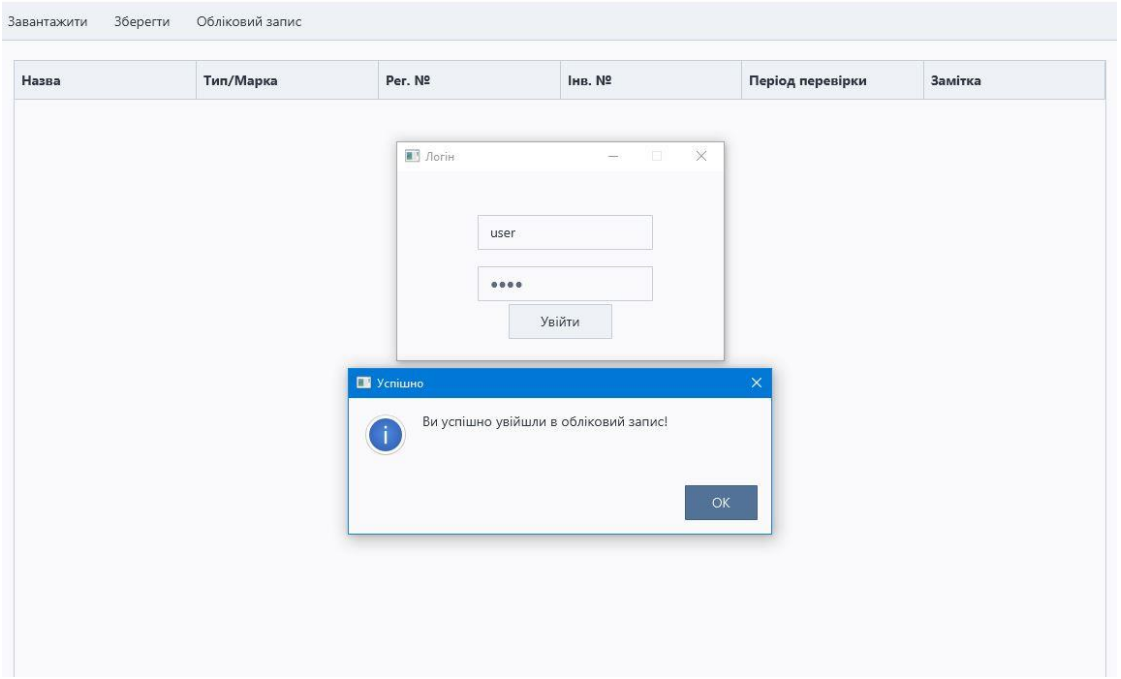

Рисунок 4.3 – Авторизація до облікового запису

Джерело: зроблено автором

Діаграма класів розробленого серверу зображена на рисунку 4.4.

Рисунок 4.4 – діаграма класів API серверу

Джерело: побудовано автором

В результаті хмарна частина програмного додатку реалізує функціонал збереження даних в хмарному середовищі та відповідає за безпечну авторизацію користувача.

4.2 Програмна реалізація системи сповіщень

Для реалізації системи сповіщень користувача було обрано популярний в Україні месенджер Telegram. Він надає широкий та зручний функціонал для створення ботів з допомогою Telegram Bot API [33]. Враховуючи що система розробляється на мові Java було обрано відповідну відкриту бібліотеку для розробки Telegram ботів мовою Java – TelegramBots.

В нашому випадку бот буде сповіщавати користувача про потребу в повірці кожного доданого приладу за місяць до настання терміну повірки. Враховуючи функціональні вимоги до сповіщень Telegram бота буде написано з використанням Polling [34] режиму роботи. Тобто сервіс сам буде періодично опитувати сервери Telegram чи не написав користувач нову команду. Такий підхід буде мати недолік в нижчий швидкості відклику, проте в нашому випадку користувач не надсилатиме команд боту тому цей недолік не є критичним. Також Polling режим роботи має перевагу в легкості розробки в порівнянні з більш швидким проте значно складнішим Webhook [34] режимом.

Для додавання облікового запису Telegram користувача необхідно натиснути кнопку «Відкрити Telegram» в розділі обліковий запис. Далі автоматично відкриється браузер з запрошенням розпочати діалог з ботом як показано на рисунку 4.5.

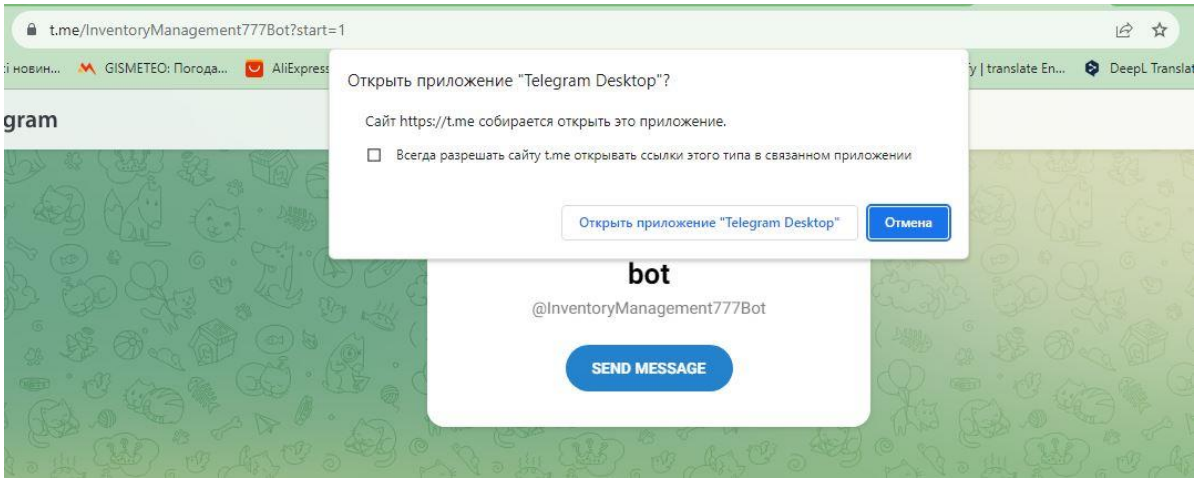

Рисунок 4.5 – запрошення на додавання Telegram бота

Джерело: зроблено автором

При початку діалогу з ботом він автоматично проінформує користувача про початок роботи, в подальшому бот буде інформувати користувача про потребу в перевірці доданих приладів. Приклад інформування про потребу в перевірці зображено на рисунку 4.6.

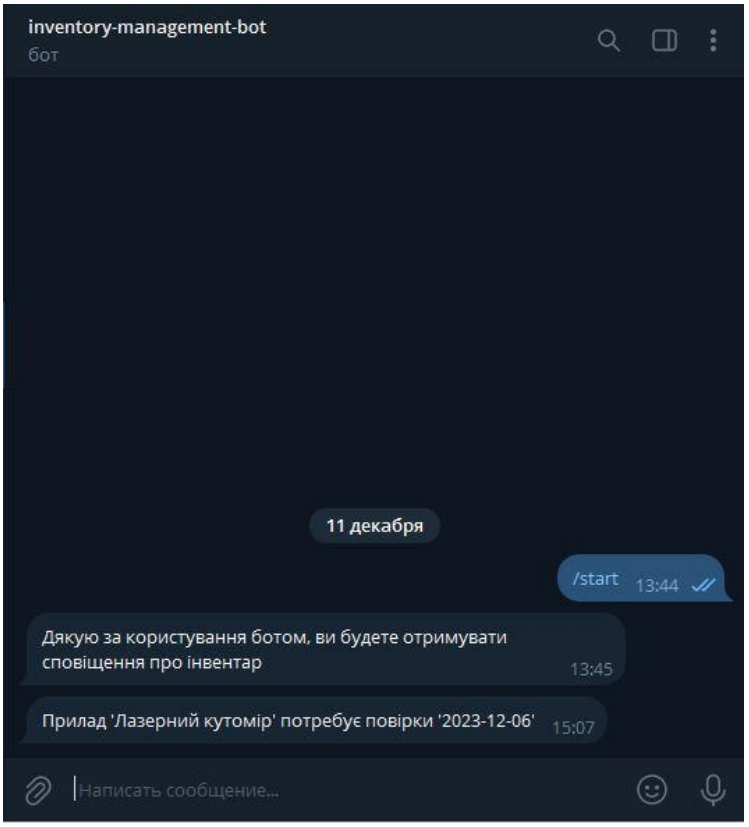

Рисунок 4.6 – інформування про потребу в перевірці

Джерело: зроблено автором

Розроблена система інформування про вичерпання термінів повірки допоможе полегшити процес ведення обліку контрольно вимірювальних приладів.

4.3 Програмна реалізація десктопної частини

Для розробки програмного додатку було використано мову програмування Java, графічний фреймворк JavaFX та додаткові бібліотеки. В основі архітектури розроблюваного додатку лежить шаблон проектування MVC, особливості цієї архітектури ми розглядали раніше тому відзначимо реалізацію частин представлення, моделі та контролера.

Даними якими оперує додаток є основними сутностями бази даних, тобто прилад, повірка та ремонт. Приклад розробки моделі для сутності «прилад» зображено на рисунку 4.7.

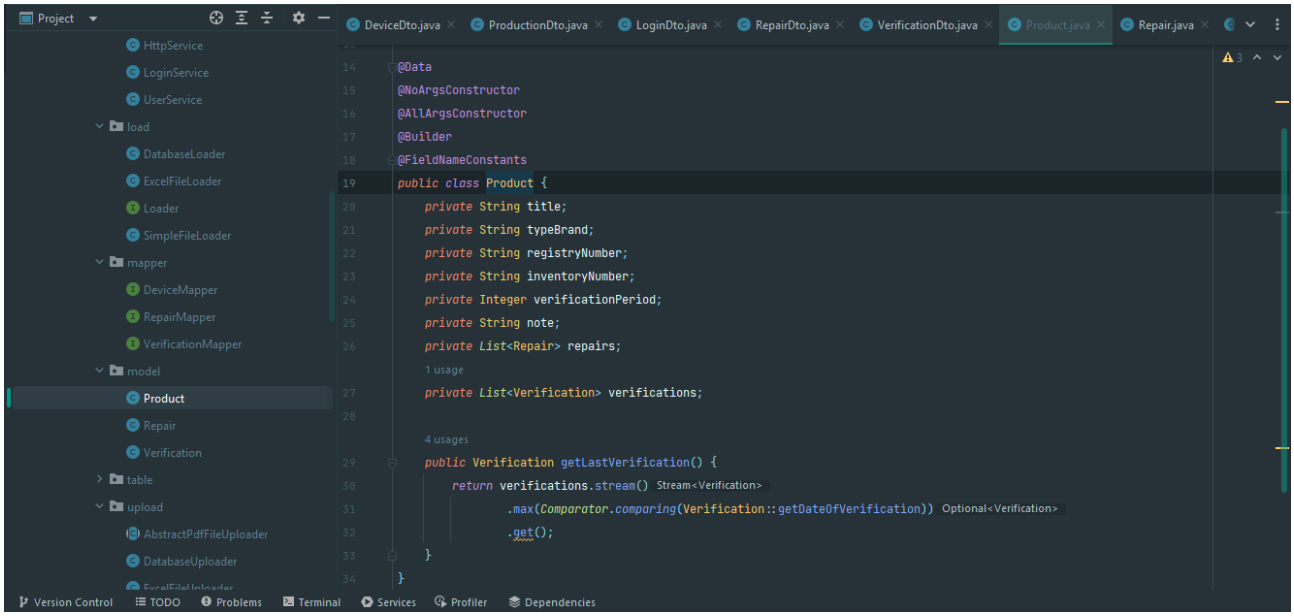

Рисунок 4.7 – Розробка моделі для сутності «прилад»

Джерело: зроблено автором

В свою чергу контролер оперує даними з моделі та реалізує певний функціонал. Кожна функція реалізована в контролері присвоюється до події в шаблонах форм FXML та помічається відповідною нотацією «@FXML». На рисунку 4.8 можемо побачити приклад коду контролеру головної сторінки додатку.

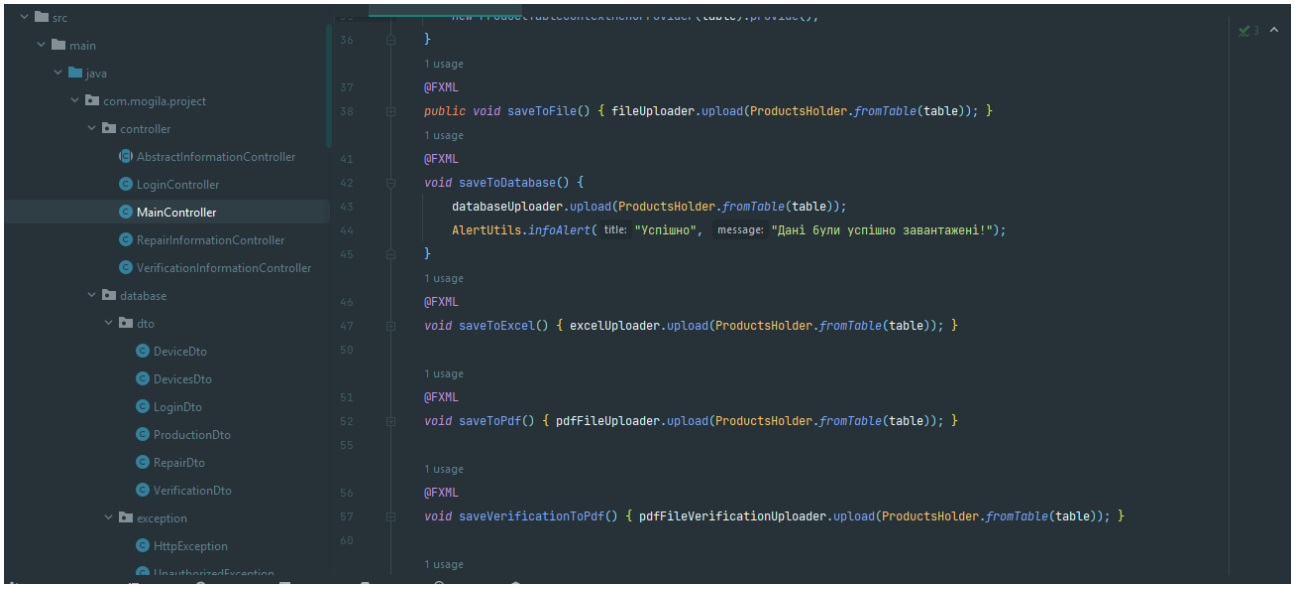

Рисунок 4.8 – Головний контролер додатку

Джерело: зроблено автором

Для зручної побудови інтерфейсу додатку було використано Scene Builder, який має зручний інтерфейс для проектування інтерфейсу заощаджуючи час на написання XML розмітки сторінок вручну, приклад інтерфейсу Scene Builder зображено на рисунку 4.9.

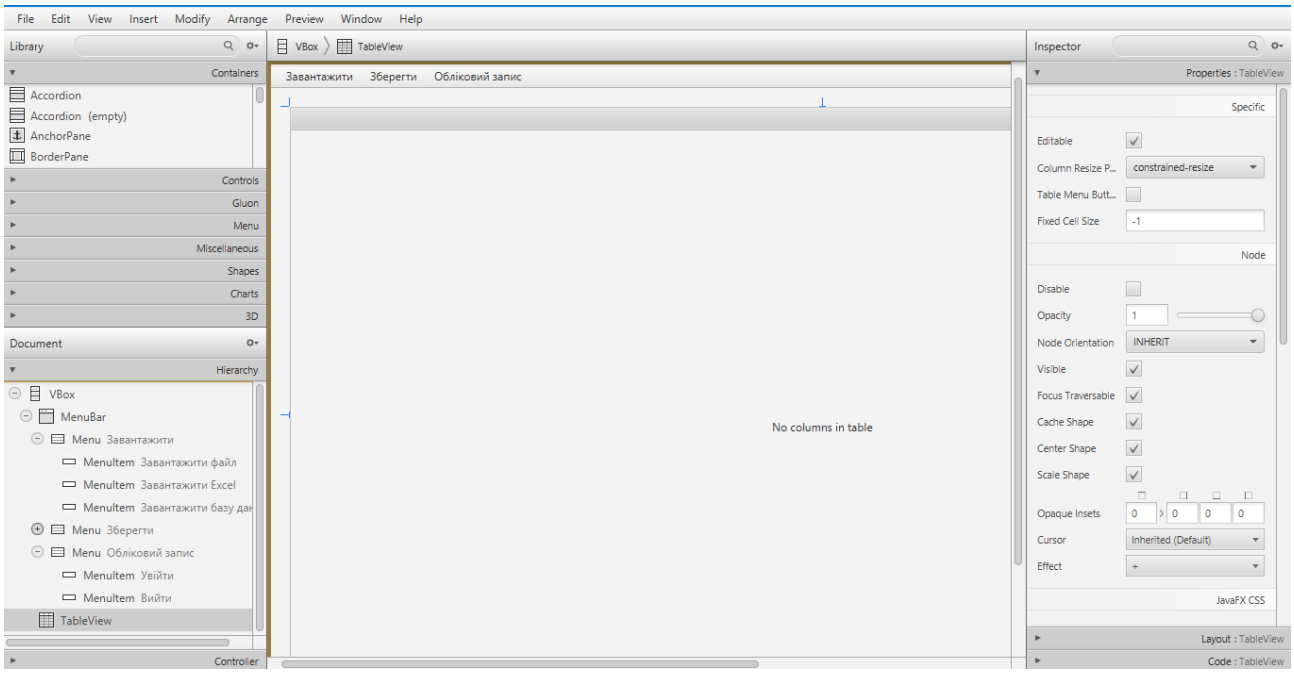

Рисунок 4.9 – створення головної сторінки додатку в Scene Builder

Джерело: зроблено автором

При розробці клієнт-серверного додатку виникає потреба в мінімізації витрачених ресурсів на обмін даними. Для вирішення проблеми нераціонального використання ресурсів в процесі обміну даними використовують шаблон проектування Data Transfer Object (DTO) [35]. Цей шаблон зменшує кількість викликів між клієнтом та сервером за рахунок створення уніфікованих класів моделі, які б іншому випадку передавались в декілька запитів. Приклад DTO класу можемо побачити на рисунку 4.10.

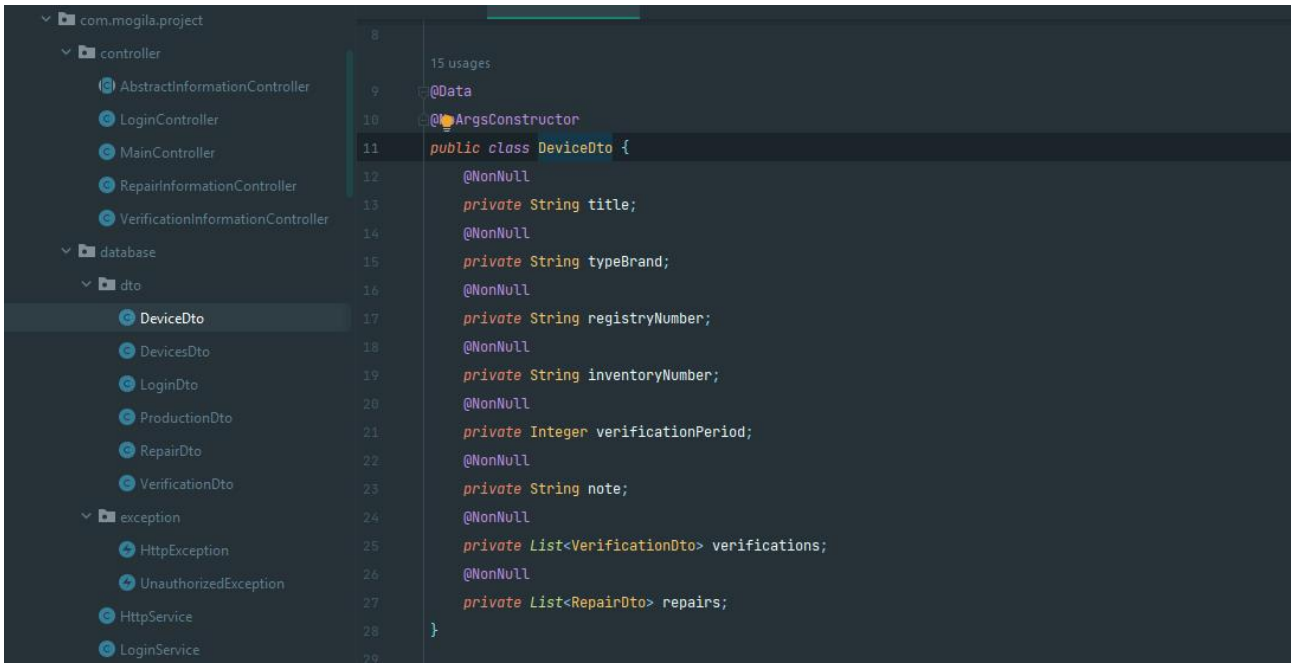

Рисунок 4.10 – Створення DTO класів

Джерело: зроблено автором

Для роботи зі списком приладів в інтерфейсі використовується функціональний елемент «TableView», для виконання функціоналу програмного додатку було обрано два типи взаємодії, це спливаючі шторки з кнопками «MenuBar» та натискання правої кнопки миші в області «TableView». Перелік функцій при натисканні ПКМ можна побачити на рисунку 4.11.

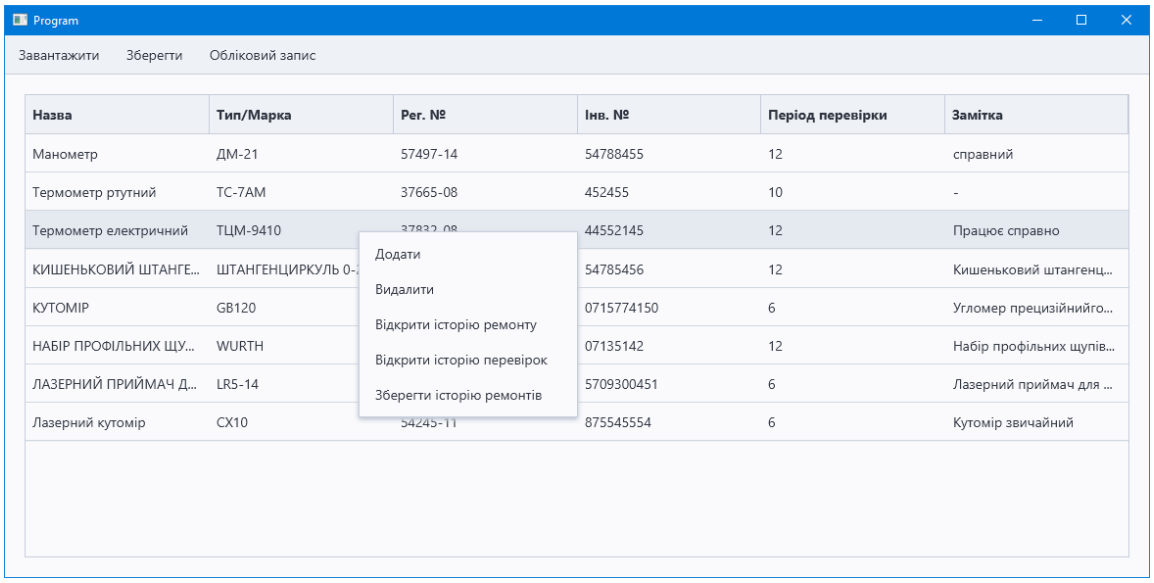

Рисунок 4.11 – Функціонал взаємодії з переліком приладів

Джерело: зроблено автором

Для кожного прибору користувач може окремо відкрити історію перевірок та ремонтів та взаємодіяти з ними: додавати, редагувати чи видаляти. На рисунку 4.12 представлено можливість роботи з історією повірок та ремонтів в окремих вікнах.

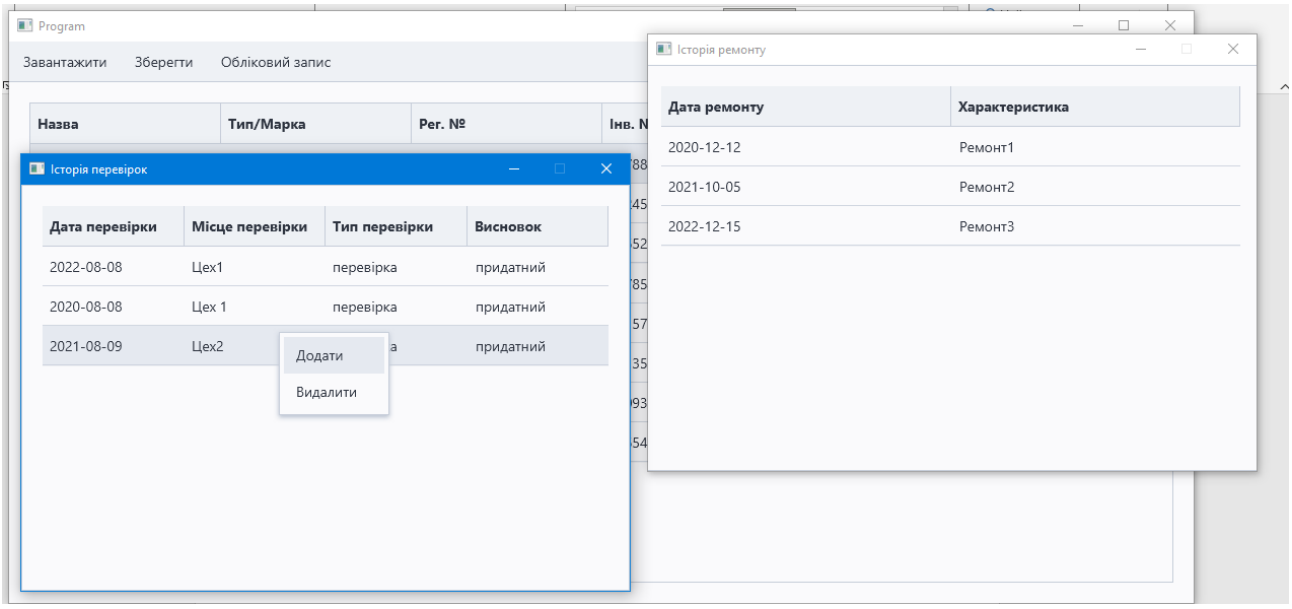

Рисунок 4.12 – Робота з полями повірок та ремонтів

Джерело: зроблено автором

Переглянути повний перелік класів розробленого програмного додатку можна на рисунку 4.13.

Рисунок 4.13 – діаграма класів програмного додатку

Джерело: побудовано автором

Розроблений програмний додаток відповідає заявленим вимогам у простоті та зручності інтерфейсу та може бути запущений на різних операційних системах на яких можливе встановлення Java.

4.4 Реалізація функціоналу ведення звітності

Для розробки функціоналу ведення звітності було використано API Apache POI [36] та бібліотеку iText [37]. За взаємодію з Excel файлами відповідає Apache POI який є потужним рішенням для організації роботи з файлами документів Microsoft Office. В свою чергу для генерації звіту в форматі PDF було використано бібліотеку iText адже попереднє рішення від Apache не підтримує роботу з цим форматом. На рисунку 4.14 зображено фрагмент класу який використовує Apache POI для реалізації завантаження даних з документу Excel.

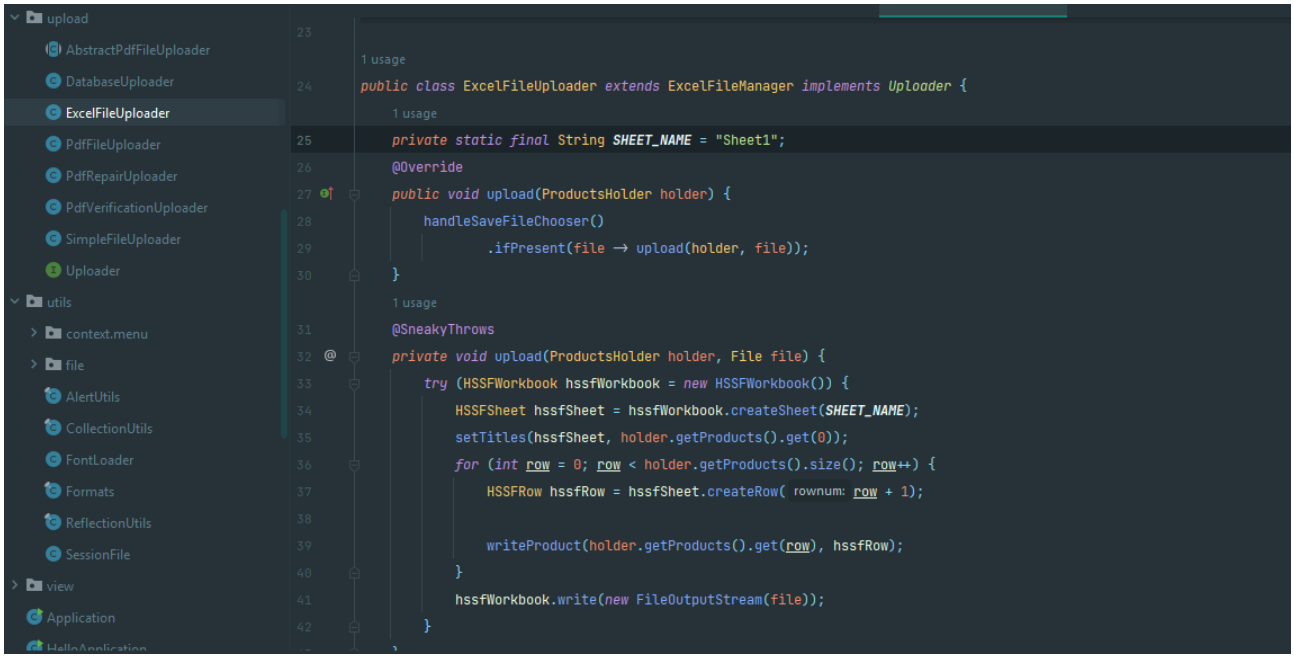

Рисунок 4.14 – Реалізація завантаження с Excel

Джерело: зроблено автором

Користувач додатку може зберегти три типи звітів: список деталей, журнал повірки та історію ремонтів. При генерації журналу повірок створюється документ з повним списком всіх приладів, що є на обліку та у відповідних колонках вказується остання дата перевірки данні про неї та прораховується наступна перевірка за планом, на рисунку 4.15 зображено приклад звіту журналу повірок в форматі PDF.

Графік

повірки контрольно-вимірювальних приладів

Тестове підприємство

#	Назва	Інв. №	Замітка	Дата ост. перевірки	Місце перевірки	Тип перевірки	Наступна перевірка
1	Манометр	54788455	справний	2023-08-09	Цех2	перевірка	2024-08-09
2	Термометр ртутний	452455	-	2023-02-18	Цех 5	Огляд	2023-12-18
3	Термометр електричний	44552145	Працює справно	2023-10-10	Цех 2	норматив 1	2024-10-10
4	КИШЕНЬКОВИЙ ШТАНГЕНЦИРКУЛЬ	54785456	Кишеньковий штангенциркуль з фіксуючим гвинтом у верхній частині 0-200мм	2023-05-03	Цех2	Огляд	2024-05-03
5	КУТОМІР	0715774150	Угломер прецизійнийгоніометрія-MATTVERCHROMT-GB120-L150	2023-10-10	Цех 6	повна перевірка	2024-04-10
6	НАБІР ПРОФІЛЬНИХ ЩУПІВ	07135142	Набір профільних щупів Wurth 20шт 0,05-1,00мм WurthFELRGAU-20PCS	2023-01-13	Цех2	огляд	2024-01-13
7	ЛАЗЕРНИЙ ПРИЙМАЧ ДЛЯ ЛАЗЕРНОГО НІВЕЛІРА	5709300451	Лазерний приймач для лазерного нівеліра LR5-14ЛАЗЕР-ПРИЙМАЛЬНИК-LR5-14	2023-11-11	Цех2	Тест	2024-05-11
8	Лазерний кутомір	875545554	Кутомір звичайний	2023-06-06	Цех6	Звичайна	2023-12-06

Рисунок 4.15 – Приклад журналу повірок

Джерело: зроблено автором

Для кожного приладу окремо можна згенерувати історію ремонтів приклад якої наведено на рисунку 4.16.

Керівник _____ Підпис _____ Дата _____

Історія

повірок контрольно-вимірювальних приладів

Манометр

Тестове підприємство

#	Дата ремонту	Характеристика
1	2020-12-12	Ремонт1
2	2021-10-05	Ремонт2
3	2022-12-15	Ремонт3

Рисунок 4.16 – Приклад історії ремонтів

Джерело: зроблено автором

В додатку було реалізовано лише базовий набір функціоналу ведення звітності, проте реалізовані класи дозволяють легко масштабувати цей набір під конкретне підприємство враховуючи його статут по оформленню та потреби.

4.5 Тестування розробленого функціоналу

Для гарантування надійної та безперебійної роботи програмного забезпечення процес розробки повинен закінчуватись тестуванням. Тестування дозволяє виявляти та усувати помилки, забезпечуючи стабільну та надійну роботу програм, відповідно до вимог користувачів, тестування також сприяє вдосконаленню функціональності та забезпеченню високого рівня задоволення від використання додатком.

В процесі розробки програмного додатку було передбачено та підготовлено обробку більшості виключних ситуацій. Також було проведено функціональне тестування [38], тобто чи вірно відпрацьовують функції додатку в нормальних умовах. На рисунках 4.17 – 4.18 зображено приклади обробки виключних ситуацій в програмного додатку.

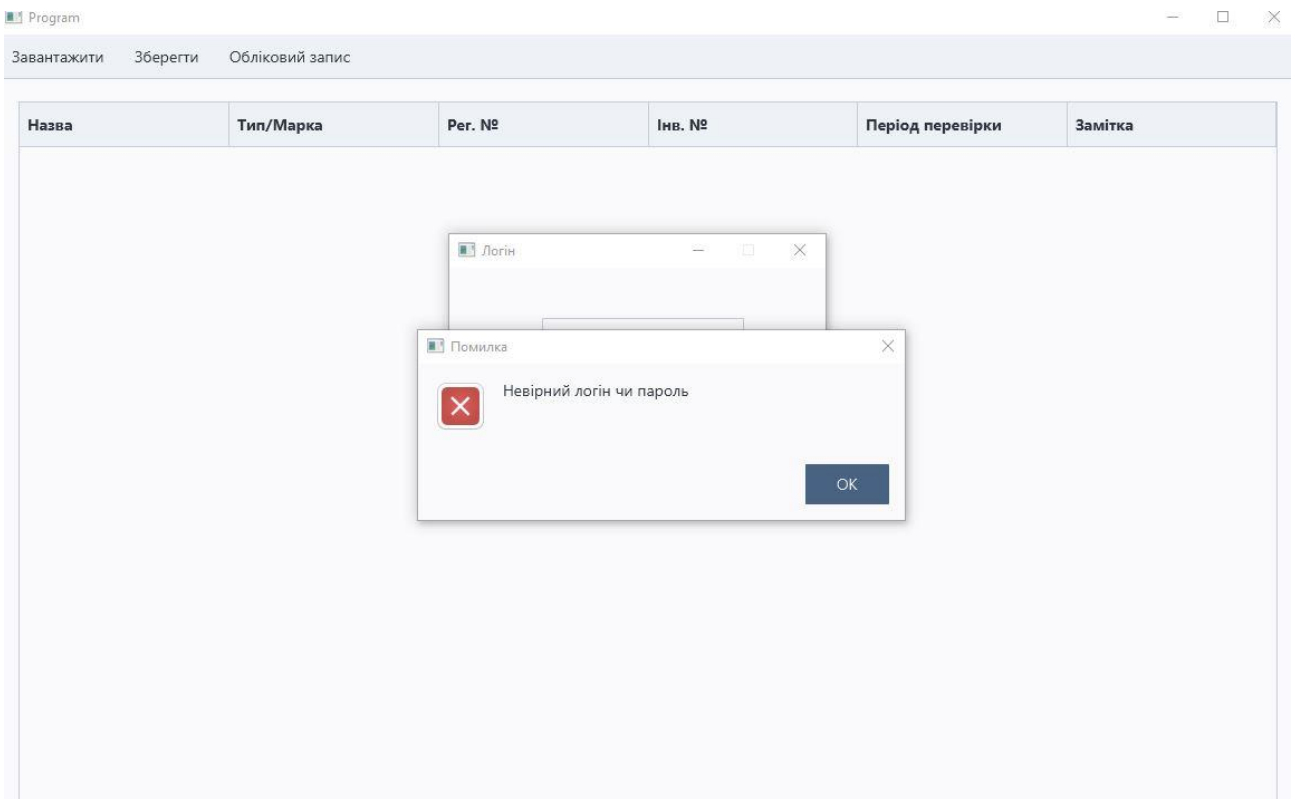

Рисунок 4.17 – Результат введення невірних авторизаційних даних

Джерело: зроблено автором

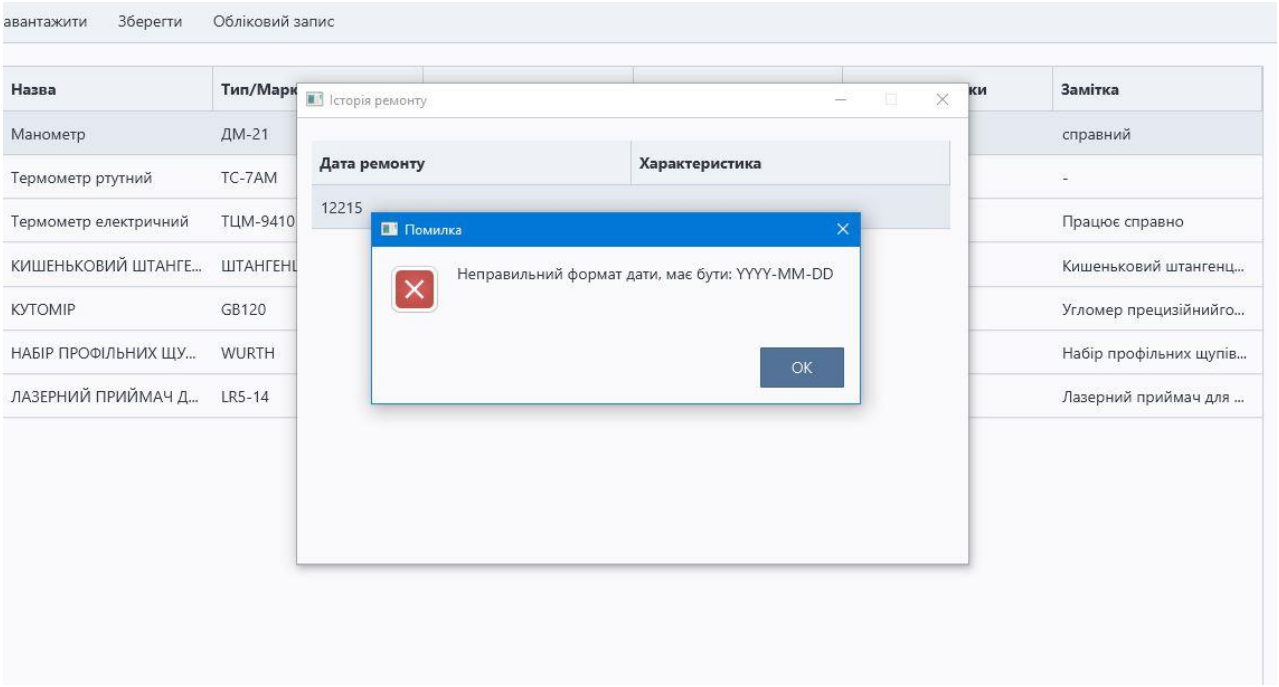

Рисунок 4.18 – Результат некоректно введеної дати

Джерело: зроблено автором

В результаті проведеного функціонального тестування та тестування методом чорного ящика було підтверджено коректність роботи функціоналу, та перевірено коректність роботи додатку в непередбачених ситуаціях. Тому можна зробити висновок, що програмний додаток ведення обліку-інвентаризації контрольно-вимірювальних приладів з марним зберіганням даних відповідає заявленим функціональним вимогам та коректно обробляє виключні ситуації.

ВИСНОВОК

В ході виконання кваліфікаційної роботи магістра було проаналізовано предметну область для розробки програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

В ході аналізу предметної області було проведено аналіз останніх досліджень і публікацій. Також провели порівняння аналогів представлених на ринку, визначено їх переваги та недоліки. Проаналізовано сучасні архітектурні підходи в проектуванні систем та обрав доречний для розробки системи. В результаті було поставлено мету та визначено задачі переддипломної практики.

В проектній частині переддипломної практики було проведено структурно функціональне моделювання з використанням діаграм Use-case та IDEF0, а також визначено список сутностей бази даних та побудовано діаграму потоків даних.

В практичній частині кваліфікаційної роботи розроблено десктопну та хмарну частину програмного додатку, реалізовано та протестовано його функціонал.

Проведено повний список планувальних робіт, визначено зміст та ієрархічну структуру робіт. В завершенні було проведено комплексний аналіз ризиків, передбачено можливі ризики в процесі роботи над проектом та шляхи їх вирішення (додаток А).

В результаті виконання роботи було розроблено програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних, використання якого сприятиме підвищенню ефективності інвентаризаційних робіт в сфері метрології.

Лістинг основних модулів розробленого програмного додатку представлено у додатку Б.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Титенко Л. В., Д. А. М. До питання вибору програмного забезпечення для організації обліку на підприємстві. .

2. Грабчук, І. Л., Ляхович, Г. І. Програмне забезпечення для ведення обліку : проблеми вибору та використання в ході аутсорсингу. 2017. Vol. 3, No. 38. С. 32–36.

3. Височан О. О. Інвентаризація: сутність, класифікація, принципи, функції та завдання: [Електронний ресурс] - Режим доступу до ресурсу: http://www.visnyk-econom.uzhnu.uz.ua/.

4. Приймак О. Ю. Інвентаризація запасів:сучасні проблеми методики та практики: [Електронний ресурс] - Режим доступу до ресурсу: http://ir.kneu.edu.ua/bitstream/handle/2010/19200/349-356.pdf?sequence=1&isAllowed=y.

5. Хорошун, Г. М. Концепції та парадигми побудови інформаційних систем та технологій в галузі оптичної метрології. Вісник Східноукраїнського національного університету імені Володимира Даля. 2023. No. 1 (277).

6. Система автоматизації обліку та управління ІС-ПРО: [Електронний ресурс] - Режим доступу до ресурсу: https://ispro.com.ua/.

7. Metrology Asset Manager: [Електронний ресурс] - Режим доступу до ресурсу: https://shop.hexagonmi.com/na/en_US/USD/Catalog/Software/Metrology-Software/Metrology-Asset-Manager/Metrology-Asset-Manager/p/H00012562.

8. Richardson, C. Microservices Patterns: *Online*. 2017.

9. What is an API (Application Programming Interface)? [Електронний ресурс] - Режим доступу до ресурсу: https://aws.amazon.com/what-is/api/.

10. Мікросервісна архітектура для початківців: [Електронний ресурс] - Режим доступу до ресурсу: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/.

11. Client server architecture: [Електронний ресурс] - Режим доступу до

ресурсу: https://www.enjoyalgorithms.com/blog/client-server-architecture.

12. Top 5 Programming Languages For Desktop App Development in 2023: [Електронний ресурс] - Режим доступу до ресурсу: https://www.decipherzone.com/blog-detail/desktop-app-programming-language.

13. Муляр, В. Розробка JavaFX-додатків із використанням Scene Builder. Computer-integrated technologies: education, science, production. 2020. No. 39.

14. Korunović, A., Vlajić, S. Example of Integration of Java GUI Desktop Technologies Using the Abstract Factory Pattern for Education Purposes. ETF Journal of Electrical Engineering. 2023. Vol. 29, No. 1.

15. Database programming with JDBC and JAVA. Computers & Mathematics with Applications. 1997. Vol. 34, No. 10.

16. JDBC API and JDBC Drivers: Oracle Database Programming with Java: Ideas, Designs, and Implementations. 2022.

17. Juba, S., Vannahme, A., Volkov, A. Learning PostgreSQL: *Pack Publishing*. 2015.

18. Klimek, B. M., Skublewska-Paszkowska, M. Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application Porównanie wydajności relacyjnych baz danych PostgreSQL oraz MySQL dla aplikacji desktopowej. JCSI. 2021. Vol. 18.

19. Klimek, B., Skublewska-Paszkowska, M. Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application. Journal of Computer Sciences Institute. 2021. Vol. 18.

20. Sarker, I. H., Apu, K. MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application. International Journal of Hybrid Information Technology. 2014. Vol. 7, No. 5.

21. Deinum, M., Cosmina, I. Pro Spring MVC with WebFlux: *Pro Spring MVC with WebFlux*. 2021.

22. Suriya, D. S., S., N. Design of UML Diagrams for WEBMED - Healthcare Service System Services. EAI Endorsed Transactions on e-Learning. 2023. Vol. 8, No. 1.

23. Мова UML. Діаграма використання: [Електронний ресурс] - Режим доступу до ресурсу: http://p4ilka.blogspot.com/2018/12/uml.html.

24. Методологія IDEF0: [Електронний ресурс] - Режим доступу до ресурсу: https://studme.org/87184/ekonomika/metodologiya_idef0.

25. Waissi, G. R., Demir, M., Humble, J. E., та ін. Automation of strategy using IDEF0 - A proof of concept. Operations Research Perspectives. 2015. Vol. 2.

26. ДСТУ 9119:2021 Метрологія. Мости, потенціометри автоматичні, самописні одноканальні, регулювальні та регулятори температури. Методика повірки: [Електронний ресурс] - Режим доступу до ресурсу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=96267.

27. Діаграма розгортання і правила її побудови. .

28. Selamat, N., Ibrahim, R. Similarity syntax rules between DFD and UML diagrams. International Journal of Advanced Trends in Computer Science and Engineering. 2019. Vol. 8, No. 3.

29. Geofann Nerissa Arviana. Data Flow Diagram (DFD): Definisi, Fungsi, dan Simbol yang Digunakan / 2021.

30. Duarte, A. Spring Boot: Practical Vaadin. 2021.

31. Varanasi, B., Bartkov, M. Spring REST: Building Java Microservices and Cloud Applications: *Spring REST: Building Java Microservices and Cloud Applications*. 2021.

32. Interface HttpSession: [Електронний ресурс] - Режим доступу до ресурсу: https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/servlet/http/HttpSession.html.

33. Santoso, W., Nurjannah, W., Shudhuashar, M., та ін. The Development of Telegram Bot Api to Maximize The Dissemination Process of Islamic Knowledge in 4.0 Era. Jurnal teknik informatika. 2022. Vol. 15, No. 1.

34. Long Polling vs. Webhooks: [Електронний ресурс] - Режим доступу до ресурсу: https://grammy.dev/guide/deployment-types.

35. The DTO Pattern (Data Transfer Object): [Електронний ресурс] - Режим доступу до ресурсу: https://www.baeldung.com/java-dto-pattern.

36. Adrian, A. Read, write, format Excel 2007 (xlsx) files. R Guide. 2014. Vol. 2007.

37. Lowagie, B. iText in Action: *Annals of surgery*. 1955.

38. Функціональне та нефункціональне тестування: розгляд аспектів та порівняння: [Електронний ресурс] - Режим доступу до ресурсу: https://mate.academy/blog/qa/functional-non-functional-testing/.

39. Organ, C., Bottorff, C. Work Breakdown Structure (WBS) in Project Management. Review of International Comparitive Management. 2022. Vol. 17, No. 1.

40. Kukushkin, A., Zykov, S. The Dynamic Modeling of the Project Management Process. Procedia Technology. 2013. Vol. 9.

ДОДАТОК А

Планування робіт

А.1 Ідентифікація мети ІТ-проекту

Інформаційні системи є невід'ємною складовою будь-якого сучасного підприємства. Проте серед підприємств з потребою в цифровізації ведення обліку контрольно-вимірювальних приладів існує попит на розробку програмного рішення. Також необхідно відзначити, що поширені програмні додатки не використовують переваги інтернету та хмарного збереження даних. Використання інтернету значно підвищить функціональність розробленого рішення, адже надасть переваги децентралізації роботи в системі та зменшить ризики втрати інформації.

Отже, метою проектної роботи є розробка програмного додатку для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

Для попередження помилок в розробці проекту на концептуальному рівні необхідно провести його деталізацію з використанням SMART-методу. Результати деталізації методом SMART розміщено у таблиці А.1.

Таблиця А.1 – Деталізація мети проекту методом SMART

Джерело: побудовано автором

Specific (конкретна)	Розробити програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.
Measurable (вимірювана)	Розроблений програмний додаток для ведення обліку.

Продовження табл. А1

Achievable (досяжна, узгоджена)	Для виконання проекту наявні необхідні знання Java, JavaFX, баз даних PostgreSQL та навичок написання документації. Враховуючи доступні ресурсні можливості та обмеження мета є такою, яку можливо досягти.
Relevant (реалістична)	Створений програмний додаток значно полегшить процес ведення обліку та інвентаризації контрольно-вимірювальних приладів.
Time-framed (обмежена в часі)	Ціль має часове обмеження. Термін досягнення мети проекту визначено за домовленістю між замовником та виконавцем і складає один місяць.

А.2 Планування змісту структури робіт ІТ-проекту

Ієрархічна структура робіт (WBS) [39] - це графічне представлення всіх робіт, необхідних для виконання проекту. Вона організовує всі роботи в ієрархію, починаючи з продукту проекту і закінчуючи елементарними роботами.

WBS має кілька переваг. По-перше, вона допомагає в організації командної роботи, оскільки чітко визначає, хто відповідає за виконання кожної роботи. По-друге, WBS забезпечує основу для оцінки термінів та витрат на проект. По-третє, вона допомагає в управлінні змінами, оскільки чітко визначає, які роботи необхідно змінити, якщо проект змінюється.

WBS складається з декількох рівнів. На найвищому рівні розміщений продукт проекту. На другому рівні розміщені основні дії та заходи, необхідні для досягнення мети проекту. Декомпозиція робіт продовжується до тих пір, поки всі роботи не стануть елементарними.

Елементарні роботи - це найпростіші роботи, які можна виконати. Вони мають наступні характеристики:

- Вони мають однозначний чіткий результат.
- Вони можуть бути виконані однією конкретною особою.
- Для них можна обчислити витрати праці і тривалість виконання.

На рисунку А.1 представлено WBS з розробки програмного додатку ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних.

Наступним етапом після декомпозиції робіт є створення організаційної структури виконавців (OBS) [40].

OBS - це графічне представлення людей, які беруть участь у виконанні проекту. Вона показує, хто відповідає за які роботи.

У ролі відповідальних осіб виступають співробітники, які відповідають за виконання елементарних робіт, зазначених у WBS.

Елементарні роботи - це найпростіші роботи, які можна виконати. Вони мають однозначний чіткий результат, їх може виконати одна конкретна особа, і для них можна обчислити витрати праці і тривалість виконання.

Кожну елементарну роботу можна розглядати як окремий проект.

Це означає, що для кожної елементарної роботи можна розробити свій план, бюджет і команду.

На рисунку А.2 представлено організаційну структуру планування проекту. Список виконавців, що функціонують в проекті описано в таблиці А.2.

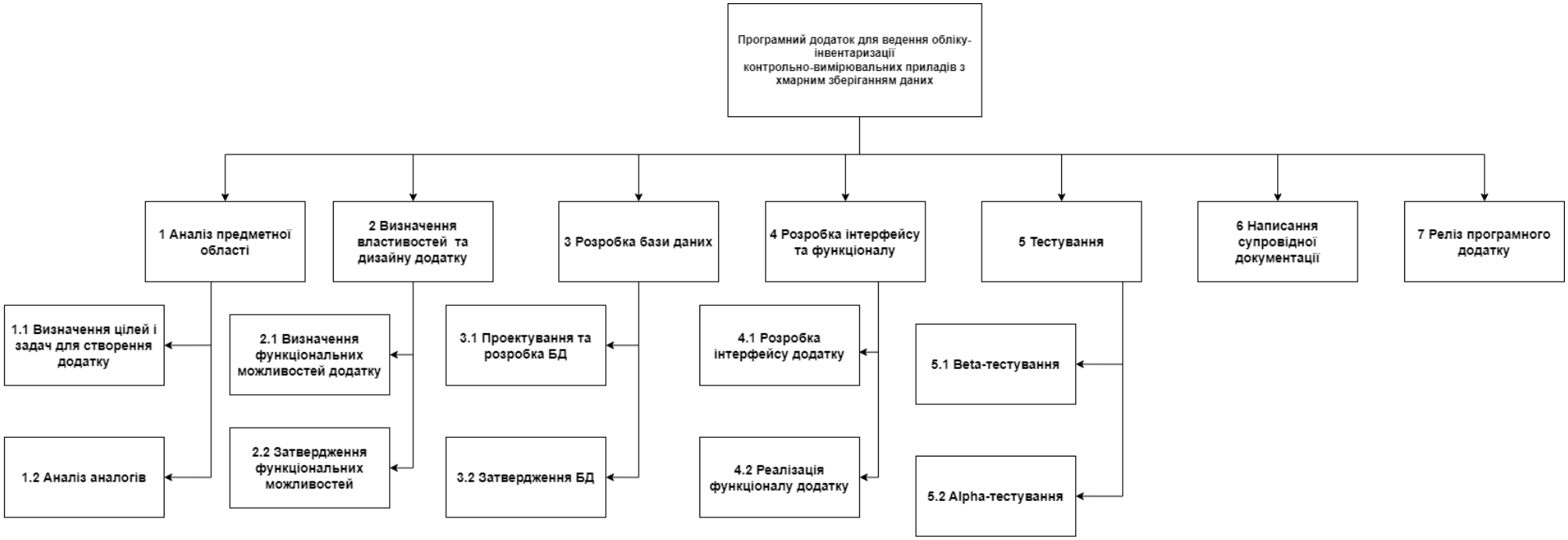

Рисунок А.1 – WBS-структура робіт проекту

Джерело: зроблено автором

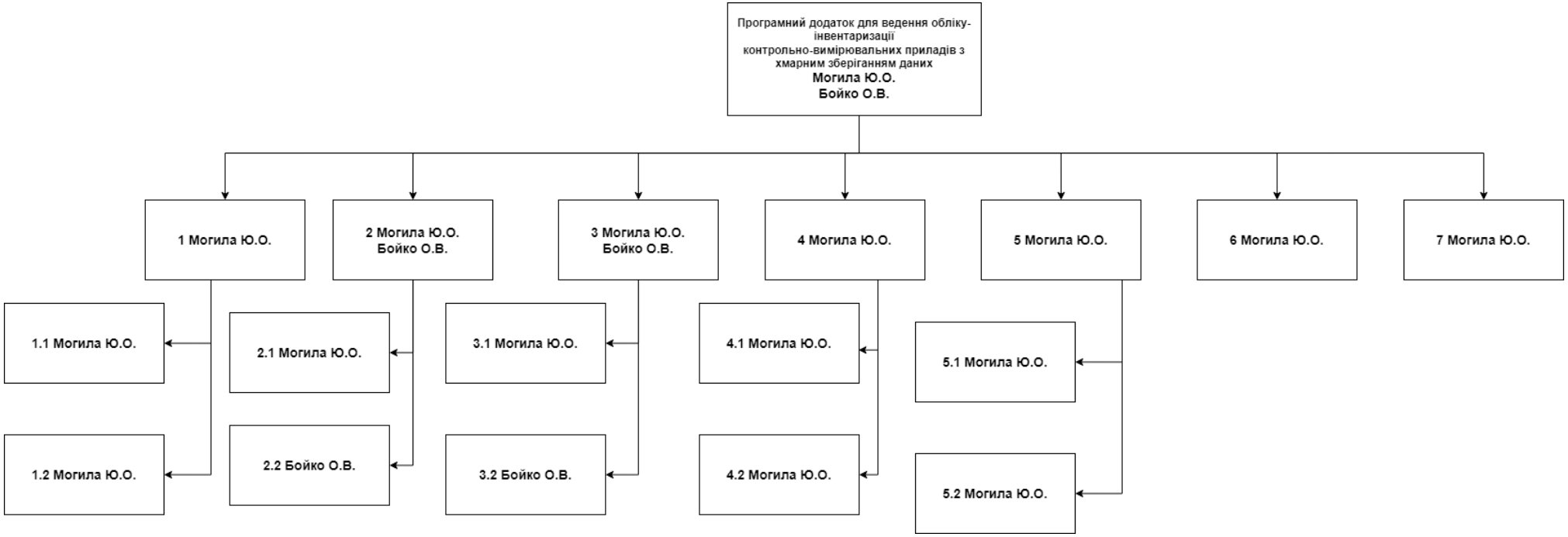

Рисунок А.2 – OBS-структура робіт проект

Джерело: зроблено автором

Таблиця А.2 – Виконавці проекту

Джерело: побудовано автором

Роль	**Ім'я**	**Проектна роль**
Розробник	Могила Ю.О.	Виконує розробку програмного додатку.
Проектувальник	Могила Ю.О.	Виконує проектування бази даних та розробляє структуру програмного додатку.
Тестувальник	Могила Ю.О.	Відповідає за тестування функціоналу та дизайну програмного додатку.
Керівник проекту	Бойко О.В.	Формує завдання на розробку проекту.
Менеджер проекту	Могила Ю.О.	Відповідає за виконання термінів, розподіл ресурсів. Виконує збір та аналіз даних.

А.3 Побудова календарного графіку виконання ІТ - проекту

Побудова календарного графіку (діаграми Ганта) є важливим етапом планування проекту.

Ця діаграма показує, коли будуть виконуватися кожна робота в проекті. Вона допомагає в оцінці тривалості проекту та в забезпеченні того, щоб всі роботи були виконані вчасно.

Діаграма Ганта виглядає як розклад виконання робіт з реальним розподілом дат.

Кожна робота на діаграмі представлена в вигляді вертикальної смуги. Ширина смуги відповідає тривалості роботи. Дати початку та закінчення роботи вказуються на шкалі часу, яка розташована горизонтально.

Завдяки діаграмі Ганта можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят.

Діаграма Ганта враховує такі фактори, як:

- Тривалість кожної роботи
- Доступні ресурси
- Вихідні дні та свята

Це дозволяє отримати достовірне уявлення про те, коли буде завершено проект.

Календарний графік проекту представлено на рисунку А.3.

	Режи задач	Название задачи	Длитель	Начало	Окончание	Предшественники	Названия ресурсов	авить новый стол
		Програмний додаток для ведення обліку-інвентаризації контрольно-вимірювальних приладів з хмарним зберіганням даних	**31 дней?**	**Пн 06.11.23**	**Вс 17.12.23**			
		Аналіз предметної області	**4 дней**	**Пн 06.11.23**	**Чт 09.11.23**			
		Визначення цілей і задач для створення додатку	2 дней	Пн 06.11.23	Вт 07.11.23			
		Аналіз аналогів	2 дней	Ср 08.11.23	Чт 09.11.23	3		
		Визначення властивостей та дизайну додатку	**5 дней?**	**Чт 09.11.23**	**Ср 15.11.23**	**2**		
		Визначення функціональних можливостей додатку	2 дней	Пт 10.11.23	Пн 13.11.23	4		
		Затвердження функціональних можливостей	3 дней	Вт 14.11.23	Чт 16.11.23	6		
		Розробка бази даних	**5 дней?**	**Чт 16.11.23**	**Ср 22.11.23**	**5**		
		Проектування та розробка БД	4 дней	Пт 17.11.23	Ср 22.11.23	7		
		Затвердження БД	1 день	Чт 23.11.23	Чт 23.11.23	9		
		Розробка інтерфейсу та функціоналу	**10 дней?**	**Чт 23.11.23**	**Ср 06.12.23**	**8**		
		Розробка інтерфейсу додатку	5 дней	Пт 24.11.23	Чт 30.11.23	10		
		Реалізація функціоналу додатку	5 дней	Пт 01.12.23	Чт 07.12.23	12		
		Тестування	4 дней?	Чт 07.12.23	Вт 12.12.23	**11**		
		Beta-тестування	2 дней	Пт 08.12.23	Пн 11.12.23	13		
		Alpha-тестування	2 дней	Вт 12.12.23	Ср 13.12.23	15		
		Написання супровідної документації	2 дней	Ср 13.12.23	Чт 14.12.23	14		
		Реліз програмного додатку	1 день	Пт 15.12.23	Вс 17.12.23	17		

Рисунок А.3 – Календарний графік проекту

Джерело: зроблено автором

А.4 Планування ризиків проекту

Під час виконання якісної оцінки ризиків проекту необхідно визначити та виділити ті ризики, які вимагають негайного усунення. Ефективність реакції на ризик буде визначатися ступенем його важливості. Управління ризиками проекту включає в себе вчасне та адекватне реагування на ідентифіковані небезпеки.

На наступному етапі виконується кількісне оцінювання ризиків, що надає можливість кількісно оцінити вплив ризиків на проект та визначити оптимальні стратегії управління ними. Ці два підходи до оцінювання ризиків можуть використовуватися одночасно або окремо, залежно від ступеня забезпечення проекту та його особливостей. Загальна мета полягає в тому, щоб забезпечити успішне управління ризиками та збереження стабільності проекту протягом всього його життєвого циклу. У таблиці А.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця А.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу.

Джерело: зроблено автором

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для зниження негативного впливу ризиків на проект необхідно провести процес планування реагування на них. Цей процес включає визначення ефективності стратегій вирішення та оцінку можливих наслідків для проекту. Оцінка здійснюється з використанням показників, що розглядаються в таблиці

А.3. В результаті реагування було створено матрицю ймовірності виникнення ризиків та впливу на них, яка представлена на рисунку А.4.

Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Рисунок А.4. – Матриця ймовірності

Джерело: зроблено автором

Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці А.4. У таблиці А.5 описано ризики та стратегії реагування на кожен з них.

Таблиця А.4 – Шкала оцінювання за рівнем ризику.

Джерело: зроблено автором

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$1 \leq R \leq 2$	1,5,9
2	Виправдані	$3 \leq R \leq 4$	2,7,8,10
3	Недопустимі	$6 \leq R \leq 9$	3,4,6

Таблиця А.5 – Ризики та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	3	1. Налагодити гарні відносини між розробником та керівником. 2. Дотримуватися ділового етикету спілкування. 3. Створити комфортні умови для співпраці	Попередження	При виявлення непорозуміння потрібно вияснити, що саме стало причиною непорозуміння обсудити її та створити здорову атмосферу в колективі.
RS_2	Відкритий	Низька кваліфікація розробників	Середня	Середній	4	1.Підвищити кваліфікацію персоналу. 2. Переглянути онлайн-ресурси для підвищення рівня знань.	Пом’якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки.
RS_3	Відкритий	Нечітке завдання на розробку	Середня	Високий	6	1.Ясно і однозначно обговорити із замовником усі види вимог. 2.Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3.Періодичний контроль замовником етапів роботи.	Попередження	При виявленні невідповідностей деяких характеристик продукту заявленим вимогам потрібно уважно та чітко окреслити те, що було виконано невірно та зробити правки.

Продовження таблиці А.5.

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Вибір не ефективної технології розробки	Середня	Високий	6	1.Проаналізувати методи та засоби, для виконання проекту. 2.Обрати зрозумілу та легку в використанні технологію розробки.	Пом’якшення	Виділити час та ресурсі на пошуки покращення обраної технології. Застосувати допоміжні ресурси.
RS_5	Відкритий	Неправильна оцінка в масштабі проекту	Низька	Середній	2	1.Провести детальний аналіз проекту. 2.Визначити основні етапу проекту, розподілити час на їх виконання. 3.Проаналізувати масштаби проекту на основі додаткових джерел.	Пом’якшення	Переоцінка масштабів проекту. Перебудова стратегії реалізації проекту.
RS_6	Відкритий	Помилки проектування	Висока	Високий	9	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати.	Пом’якшення	Здійснювати проміжний контроль результатів в ході виконання проекту.

Продовження таблиці А.5.

RS_7	Не ефективна технологія розробки	Низька	Середній	3	Провести аналіз методів розробки та обрати найпростішу	Пом’якшення	Проаналізувати основні методи розробки та обрати підходящий.
RS_8	Збої в роботі програми	Середня	Середній	4	Регулярне створення резервних копій.	Попередження	Зміна сервера та забезпечення
RS_9	Реалізація зайвого функціонал у	Низька	Низький	1	Слідування раніше створеній та узгодженій документації	Пом’якшення	
RS_10	Недостатня кількість тестового покриття	Середня	Середній	4	Залучення спеціалістів	Попередження	Створення тест плану та тест кейсів на початку розробки

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ

DeviceDto

```
@Data
@NoArgsConstructor
public class DeviceDto {
    @NonNull
    private String title;
    @NonNull
    private String typeBrand;
    @NonNull
    private String registryNumber;
    @NonNull
    private String inventoryNumber;
    @NonNull
    private Integer verificationPeriod;
    @NonNull
    private String note;
    @NonNull
    private List<VerificationDto> verifications;
    @NonNull
    private List<RepairDto> repairs;

    public ZonedDateTime getLastVerification() {
        return verifications.stream()
                .max(Comparator.comparing(VerificationDto::getDateOfVerification))
                .map(VerificationDto::getDateOfVerification)
                .map(localDate ->
ZonedDateTime.now(ZoneOffset.UTC).with(localDate).plusMonths(verificationPeriod))
                .get();
    }
}
Device.java
@Entity
@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
public class Device {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(nullable = false)
    private String title;

    @Column(nullable = false)
    private String typeBrand;

    @Column(nullable = false)
    private String registryNumber;

    @Column(nullable = false)
    private String inventoryNumber;

    @Column(nullable = false)
    private Integer verificationPeriod;

    @Column(nullable = false)
    private String note;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User user;
```

```
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "device_id")
    private List<Verification> verifications;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "device_id")
    private List<Repair> repairs;
}
```

AuthController

```
@RestController
@AllArgsConstructor
public class AuthController {
    private final AuthenticationManager authenticationManager;
    private final SecurityContextRepository contextRepository;
    @PostMapping("/api/login")
    public void login(@RequestBody LoginDto loginDto, HttpServletRequest request,
HttpServletResponse response) {
        final UsernamePasswordAuthenticationToken token =
                new UsernamePasswordAuthenticationToken(loginDto.getLogin(),
loginDto.getPassword());
        final Authentication authenticated = authenticationManager.authenticate(token);
     contextRepository.saveContext(SecurityContextUtils.forAuthentication(authenticated),
request, response);
    }
}
```

WebSecurityConfig

```
@Configuration
public class WebSecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)
                .formLogin(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(authorize ->
                        authorize.requestMatchers("/api/login").permitAll()
                                .anyRequest().authenticated()
                )
                .requestCache(cache -> cache.requestCache(new NullRequestCache()))
                .exceptionHandling(handling -> handling.authenticationEntryPoint(new
HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED)));

        return http.build();
    }

    @Bean
    public UserDetailsManager users(DataSource dataSource) {
        return new JdbcUserDetailsManager(dataSource);
    }

    @Bean
    public AuthenticationManager authenticationProvider(PasswordEncoder passwordEncoder,
                                                        DatabaseUserDetailsService
userDetailsService) {
        DaoAuthenticationProvider provider = new
DaoAuthenticationProvider(passwordEncoder);
        provider.setUserDetailsService(userDetailsService);

        return new ProviderManager(provider);
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
    @Bean
    public PlatformTransactionManager transactionManager(DataSource dataSource) {
```

```
        return new DataSourceTransactionManager(dataSource);
    }
}
```

LoginController

```
public class LoginController {
    @FXML
    private TextField login;

    @FXML
    private PasswordField password;

    private final LoginService loginService = new LoginService();

    @FXML
    public void doLogin() {
        if (StringUtils.isEmpty(login.getText()) ||
StringUtils.isEmpty(password.getText())) {
            AlertUtils.errorAlert("Помилка", "Заповніть всі поля");
        }

        try {
            loginService.doLogin(login.getText(), password.getText());
            AlertUtils.infoAlert("Успішно", "Ви успішно увійшли в обліковий запис!");
            password.getScene().getWindow().hide();
        } catch (UnauthorizedException e) {
            AlertUtils.errorAlert("Помилка", "Невірний логін чи пароль");
        } catch (HttpException e) {
            AlertUtils.errorAlert("Помилка", "Невдалося увійти в обліковий запис (Код
помилки " + e.getStatusCode() + ")");
        }
    }
}
```

MainController

```
public class MainController {
    @FXML
    private TableView<Product> table;

    /* uploaders */
    private final Uploader fileUploader = new SimpleFileUploader();
    private final Uploader excelUploader = new ExcelFileUploader();
    private final Uploader databaseUploader = new DatabaseUploader();
    private final Uploader pdfFileUploader = new PdfFileUploader();
    private final Uploader pdfFileVerificationUploader = new PdfVerificationUploader();

    /* loaders */
    private final Loader fileLoader = new SimpleFileLoader();
    private final Loader excelLoader = new ExcelFileLoader();
    private final Loader databaseLoader = new DatabaseLoader();

    private final TelegramService telegramService = new TelegramService();

    @FXML
    public void initialize() {
        new ProductTableInitializer(table).initialize();
        new ProductTableContextMenuProvider(table).provide();
    }

    @FXML
    public void saveToFile() {
        fileUploader.upload(ProductsHolder.fromTable(table));
    }

    @FXML
    void saveToDatabase() {
        databaseUploader.upload(ProductsHolder.fromTable(table));
```

```
        AlertUtils.infoAlert("Успішно", "Дані були успішно завантажені!");
    }

    @FXML
    void saveToExcel() {
        excelUploader.upload(ProductsHolder.fromTable(table));
    }

    @FXML
    void saveToPdf() {
        pdfFileUploader.upload(ProductsHolder.fromTable(table));
    }

    @FXML
    void saveVerificationToPdf() {
        pdfFileVerificationUploader.upload(ProductsHolder.fromTable(table));
    }

    @FXML
    public void loadFile() {
        table.setItems(fileLoader.loadFXML());
    }

    @FXML
    void loadExcel() {
        table.setItems(excelLoader.loadFXML());
    }

    @FXML
    void loadDatabase() {
        table.setItems(databaseLoader.loadFXML());
    }

    @FXML
    void doLogin() {
        new LoginForm().show();
    }

    @FXML
    void doLogout() {
        new SessionFile().delete();
        AlertUtils.infoAlert("Успішно", "Ви вйшли з облікового запису!");
    }

    @FXML
    void openTelegram() {
        telegramService.openTelegram();
    }
}
```

HttpService

```
public class HttpService {
    private final static String SERVER_URL = "http://129.159.200.195:80";

    private final ObjectMapper mapper = mapper();

    public <T> ResponseExtractor doPost(@NonNull T body, @NonNull String endpoint) {
        return doPost(body, endpoint, Collections.emptyMap());
    }

    public <T> ResponseExtractor doPost(@NonNull T body, @NonNull String endpoint,
@NonNull Map<String, String> headers) {
        HttpRequest request = httpBuilder(headers)
                .uri(buildUri(endpoint))
                .POST(buildBody(body))
                .build();

        return doRequest(request);
    }
```

```
    public ResponseExtractor doGet(@NonNull String endpoint, @NonNull Map<String, String>
headers) {
        HttpRequest request = httpBuilder(headers)
                .uri(buildUri(endpoint))
                .GET()
                .build();

        return doRequest(request);
    }

    private ResponseExtractor doRequest(HttpRequest request) {
        HttpResponse<String> response = doInternalRequest(request);
        handleError(response);

        return new ResponseExtractor(mapper, response);
    }

    @SneakyThrows
    private static HttpResponse<String> doInternalRequest(HttpRequest request) {
        return HttpClient.newHttpClient()
                .send(request, HttpResponse.BodyHandlers.ofString());
    }

    @SneakyThrows
    private <T> HttpRequest.BodyPublisher buildBody(T body) {
        return HttpRequest.BodyPublishers.ofString(mapper.writeValueAsString(body));
    }

    private static HttpRequest.Builder httpBuilder(Map<String, String> headers) {
        HttpRequest.Builder builder = HttpRequest.newBuilder()
                .header("Content-Type", "application/json");

        headers.forEach(builder::header);

        return builder;
    }

    private static void handleError(HttpResponse<String> response) {
        int responseCode = response.statusCode();

        if (responseCode < 200 || responseCode >= 300) {
            if (responseCode == 401) {
                throw new UnauthorizedException();
            }
            throw new HttpException(responseCode);
        }
    }

    @SneakyThrows
    private static URI buildUri(String endpoint) {
        return new URI(String.format("%s%s", SERVER_URL, endpoint));
    }

    private static ObjectMapper mapper() {
        ObjectMapper objectMapper = new ObjectMapper();
        objectMapper.registerModule(new JavaTimeModule());
        objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);

        return objectMapper;
    }

    @AllArgsConstructor(staticName = "of")
    public static final class ResponseExtractor {
        private final ObjectMapper mapper;
        private final HttpResponse<String> response;

        public Optional<String> extractorHeader(@NonNull String name) {
            return response.headers().firstValue(name);
        }
```

```
        public <T> Optional<T> extractBody(Class<T> clazz) {
            return Optional.ofNullable(response.body())
                    .map(response -> extractBody(response, clazz));
        }

        public <T> Optional<List<T>> extractBodyAsList(Class<T> clazz) {
            return Optional.ofNullable(response.body())
                    .map(response -> extractBodyAsList(response, clazz));
        }

        @SneakyThrows
        private <T> T extractBody(String response, Class<T> clazz) {
            return mapper.readValue(response, clazz);
        }

        @SneakyThrows
        private <T> List<T> extractBodyAsList(String response, Class<T> clazz) {
            return mapper.readerForListOf(clazz).readValue(response);
        }
    }
}
```

DatabaseLoader

```
public class DatabaseLoader extends LoginManager implements Loader {
    private static final String ENDPOINT = "/store";

    private final HttpService httpService = new HttpService();
    private final DeviceMapper deviceMapper = new DeviceMapperImpl();

    @Override
    public List<Product> load() {
        return getSessionId()
                .map(this::load)
                .orElse(Collections.emptyList());
    }

    private List<Product> load(String session) {
        HttpService.ResponseExtractor extractor =
                httpService.doGet(ENDPOINT, Map.of(SessionFile.SESSION_HEADER_NAME,
session));

        return extractor.extractBodyAsList(DeviceDto.class)
                .map(deviceMapper::toProduct)
                .orElse(null);
    }
}
```

SimpleFileLoader

```
public class SimpleFileLoader extends SimpleFileManager implements Loader {

    @Override
    public List<Product> load() {
        return handleOpenFileChooser()
                .map(this::loadFile)
                .orElse(Collections.emptyList());
    }

    @SneakyThrows
    protected List<Product> loadFile(File file) {
        try (DataInputStream dataInputStream = new DataInputStream(new
FileInputStream(file))) {
            if (!isValid(dataInputStream)) {
                return Collections.emptyList();
            }

            return readCollection(dataInputStream, SimpleFileLoader::readProduct);
```

```
        }
    }

    @SneakyThrows
    private static <T> List<T> readCollection(DataInputStream dataInputStream,
Function<DataInputStream, T> function) {
        int sizeOfArray = dataInputStream.readInt();

        return IntStream.range(0, sizeOfArray)
                .mapToObj(integer -> function.apply(dataInputStream))
                .toList();
    }

    @SneakyThrows
    private static Product readProduct(DataInputStream dataInputStream) {
        return Product.builder()
                .title(dataInputStream.readUTF())
                .typeBrand(dataInputStream.readUTF())
                .registryNumber(dataInputStream.readUTF())
                .inventoryNumber(dataInputStream.readUTF())
                .verificationPeriod(dataInputStream.readInt())
                .note(dataInputStream.readUTF())
                .repairs(readCollection(dataInputStream, SimpleFileLoader::readRepair))
                .verifications(readCollection(dataInputStream,
SimpleFileLoader::readVerification))
                .build();
    }

    @SneakyThrows
    private static Repair readRepair(DataInputStream dataInputStream) {
        return Repair.builder()
                .dateOfRepair(toIso(dataInputStream.readUTF()))
                .characteristic(dataInputStream.readUTF())
                .build();
    }

    @SneakyThrows
    private static Verification readVerification(DataInputStream dataInputStream) {
        return Verification.builder()
                .dateOfVerification(toIso(dataInputStream.readUTF()))
                .verificationPlace(dataInputStream.readUTF())
                .typeOfVerification(dataInputStream.readUTF())
                .conclusion(dataInputStream.readUTF())
                .build();
    }

    @SneakyThrows
    private boolean isValid(DataInputStream dataInputStream) {
        if (!dataInputStream.readUTF().equals(FileMark.MARK)) {
            AlertUtils.errorAlert("Помилка", "Відкрийте правильний файл");
            return false;
        }

        return true;
    }
}
```

PdfVerificationUploader

```
public class PdfVerificationUploader extends AbstractPdfFileUploader {
    private static final List<PdfPCell> COLUMNS = List.of(
            cell("Назва"),
            cell("Інв. №"),
            cell("Замітка"),
            cell("Дата ост. перевірки"),
            cell("Місце перевірки"),
            cell("Тип перевірки"),
            cell("Наступна перевірка")
    );

    @Override
```

```
    @SneakyThrows
    protected void buildPdf(Document document, List<Product> products, String session) {
        document.add(title("Графік"));
        document.add(subTitle("повірки контрольно-вимірювальних приладів"));
        document.add(production(session));
        document.add(table(products));
    }

    @Override
    protected void customizeDocument(Document document) {
        document.setPageSize(PageSize.A4.rotate());
    }

    @SneakyThrows
    private static PdfPTable table(List<Product> products) {
        float[] widths = {
                0.2f, 0.7f, 0.7f, 0.7f, 0.7f, 0.7f, 0.7f, 0.7f
        };

        PdfPTable table = getPdfPTable(widths);
        setTitles(table);
        setProducts(table, products);

        return table;
    }

    private static void setProducts(PdfPTable table, List<Product> products) {
        MutableInt mutableInt = new MutableInt(1);

        products.stream()
                .flatMap(product -> Stream.of(
                        cell(Integer.toString(mutableInt.getAndIncrement())),
                        cell(product.getTitle()),
                        cell(product.getInventoryNumber()),
                        cell(product.getNote()),

cell(toIso(product.getLastVerification().getDateOfVerification())),
                        cell(product.getLastVerification().getVerificationPlace()),
                        cell(product.getLastVerification().getTypeOfVerification()),

cell(toIso(product.getLastVerification().getDateOfVerification().plusMonths(product.getVe
rificationPeriod())))
                ))
                .forEach(table::addCell);
    }

    private static void setTitles(PdfPTable table) {
        table.addCell(numberCell());
        COLUMNS.forEach(table::addCell);
    }
}
```

hello-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.VBox?>

<VBox alignment="TOP_CENTER" prefHeight="719.0" prefWidth="1168.0" spacing="20.0"
xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.mogila.project.controller.MainController">
    <padding>
        <Insets bottom="20.0" />
    </padding>
   <children>
      <MenuBar blendMode="DARKEN" prefWidth="568.0">
```

```
        <menus>
          <Menu mnemonicParsing="false" text="Завантажити">
            <items>
              <MenuItem mnemonicParsing="false" onAction="#loadFile" text="Завантажити
файл" />
                  <MenuItem mnemonicParsing="false" onAction="#loadExcel"
text="Завантажити Excel" />
                  <MenuItem mnemonicParsing="false" onAction="#loadDatabase"
text="Завантажити базу даних" />
            </items>
          </Menu>
          <Menu mnemonicParsing="false" text="Зберегти">
            <items>
              <MenuItem mnemonicParsing="false" onAction="#saveToFile" text="Зберегти в
файл" />
                  <MenuItem mnemonicParsing="false" onAction="#saveToExcel"
text="Зберегти в Excel" />
                  <MenuItem mnemonicParsing="false" onAction="#saveToDatabase"
text="Зберегти до бази даних" />
                  <MenuItem mnemonicParsing="false" onAction="#saveToPdf" text="Зберегти
список деталей до PDF" />
                  <MenuItem mnemonicParsing="false" onAction="#saveVerificationToPdf"
text="Зберегти журнал повірки до PDF" />
            </items>
          </Menu>
            <Menu mnemonicParsing="false" text="Обліковий запис">
               <items>
                  <MenuItem mnemonicParsing="false" onAction="#doLogin" text="Увійти" />
                  <MenuItem mnemonicParsing="false" onAction="#doLogout" text="Вийти" />
                  <MenuItem mnemonicParsing="false" onAction="#openTelegram"
text="Відкрити Telegram" />
               </items>
            </Menu>
        </menus>
      </MenuBar>
      <TableView fx:id="table" editable="true" prefHeight="660.0" prefWidth="1128.0">
         <columnResizePolicy>
            <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
         </columnResizePolicy>
         <VBox.margin>
            <Insets left="20.0" right="20.0" />
         </VBox.margin>
      </TableView>
   </children>
</VBox>
```

TableContextMenuProvider

```
@AllArgsConstructor
public abstract class TableContextMenuProvider<T> {
    private final TableView<T> tableView;

    public void provide() {
        tableView.setContextMenu(buildContextMenu(getEmptiesMenu(), new TableRow<>(),
tableView));
        tableView.setRowFactory(table -> {
            TableRow<T> row = new TableRow<>();
            bindContextMenu(row, tableView);

            return row;
        });
    }

    protected abstract List<BiFunction<TableView<T>, TableRow<T>, MenuItem>>
getEmptiesMenu();

    protected abstract List<BiFunction<TableView<T>, TableRow<T>, MenuItem>>
getMenusOnRow();

    private void bindContextMenu(TableRow<T> row, TableView<T> tableView) {
        row.contextMenuProperty().bind(
```

```
                Bindings.when(row.emptyProperty())
                        .then(buildContextMenu(getEmptiesMenu(), row, tableView))
                        .otherwise(buildContextMenu(getMenusOnRow(), row, tableView))
        );
    }

    private ContextMenu buildContextMenu(List<BiFunction<TableView<T>, TableRow<T>,
MenuItem>> items,
                                         TableRow<T> row,
                                         TableView<T> tableView) {
        ContextMenu contextMenu = new ContextMenu();

        items.stream()
                .map(func -> func.apply(tableView, row))
                .forEach(menu -> contextMenu.getItems().add(menu));

        return contextMenu;
    }
}
```

NumberProductColumn

```
public class NumberProductColumn<T> extends AbstractTableColumn<T, Integer> {
    public NumberProductColumn(@NonNull String columnName,
                               @NonNull String fieldName,
                               @NonNull BiConsumer<T, Integer> editEvent) {
        super(columnName, fieldName, editEvent);
    }

    @Override
    protected StringConverter<Integer> getConvertor() {
        return new StringConverter<>() {
            @Override
            public String toString(Integer object) {
                return Optional.ofNullable(object)
                        .map(integer -> Integer.toString(integer))
                        .orElse(null);
            }

            @Override
            @SneakyThrows
            public Integer fromString(String string) {
                return Optional.ofNullable(string)
                        .map(Integer::parseInt)
                        .orElse(null);
            }
        };
    }

    @Override
    protected Validator getValidator() {
        return value -> Optional.ofNullable(value)
                .map(string -> {
                    try {
                        Integer.parseInt(string);
                        return true;
                    } catch (NumberFormatException e) {
                        AlertUtils.errorAlert("Помилка", "Неправильний формат числа");
                        return false;
                    }
                })
                .orElse(true);
    }
}
```