Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики та комп'ютеризованих технологій
(повна назва)

Кафедра Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка компоненту MES-системи для оптимізації виробничих процесів підприємства

Виконала:
студентка 4 курсу, групи АКТСІ-20-3
Вінниченко С.О.
Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
Керівник проф. Колесник Л.В.

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

20 24 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація і комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 24 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Вінниченко Софії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка компоненту MES-системи для оптимізації виробничих процесів підприємства
затверджена наказом університету від 03 червня 2024 р. № 545 Ст
2. Термін подання студентом роботи до екзаменаційної комісії: 17.06.2024 р.
3. Вихідні дані до роботи Функція: Розробка компоненту MES-системи для оптимізації виробничих процесів підприємства. Форма діалогу: вебдодаток. Перелік програмних засобів, використовуваних в роботі: IntelliJIDEA, Java, Spring Boot, Thymeleaf, MySQL, MAMP, PhpMyAdmin, Hibernate, Bootstrap. Технічне обладнання: комп'ютер або ноутбук з веббраузером Google Chrome.
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ; 4.2 Аналіз предметної області; 4.3 Опис вимог до розроблюваного компоненту MES-системи; 4.4 Створення бази даних предметної області; 4.5 Опис прийнятих проєктних рішень; 4.6 Висновки.
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри): Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.pptx) – 14 сторінок формату А4

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, ПІБ)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання	Примітка
1	Аналіз предметної області, літератури	22.04.2024-02.05.2024	Виконано
2	Огляд компонентів MES-системи	02.05.2024-06.05.2024	Виконано
3	Визначення вимог для розробки компоненту	06.05.2024-09.05.2024	Виконано
4	Створення бази даних предметної області	09.05.2024-20.05.2024	Виконано
5	Розробка програмного забезпечення	20.05.2024-01.06.2024	Виконано
6	Оформлення пояснювальної записки	01.06.2024-11.06.2024	Виконано
7	Подання роботи на перевірку Інтернет-сервісом Strikeplagiarism	11.06.2024-12.06.2024	Виконано
8	Подання роботи на рецензію	12.06.2024-14.06.2024	Виконано
9	Подання роботи на підпис зав. кафедри	14.06.2024-17.06.2024	Виконано
10	Подання роботи до ЕК	17.06.2024	Виконано

Дата видачі завдання 22 квітня 2024 р.

Студент _____ Вінниченко С.О.
(підпис)

Керівник роботи _____ проф. Колесник Л. В.
(підпис) (посада, прізвище, ініціали)

Я, як студентка ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Дата 07.06.2024 Підпис 

РЕФЕРАТ

Пояснювальна записка: 114 с., 1 табл., 58 рис., 3 дод., 34 джерела.

ВИРОБНИЦТВО, ОПТИМІЗАЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕЛЕФОННІ АКУМУЛЯТОРИ, MES-СИСТЕМА,

Об’єкт розробки – процес виробництва телефонних акумуляторів.

Предмет розробки – автоматизована система для проєктування та оптимізації процесів виробництва.

Мета роботи – розробка компоненту MES-системи, спрямованого на оптимізацію виробничих процесів підприємства з виробництва телефонних акумуляторів.

Методи дослідження – огляд вимог підприємства для конкретної системи, моделювання процесів виробництва за допомогою UML-діаграм, створення бази даних MySQL, розробка серверної та клієнтської частини додатку з використанням мови Java та допоміжних фреймворків.

У кваліфікаційній роботі розглянуто актуальні питання за темою, запропоновано рішення з оптимізації процесів на сучасному виробництві. Спроєктовано та розроблено програмне забезпечення компоненту MES-системи для трьох робітників підприємства, перевірено його працездатність.

Розроблене програмне забезпечення може бути інтегрованим у виробництво для забезпечення його цілісності, мінімізації простоїв обладнання та підвищення ефективності роботи підприємства в цілому.

ABSTRACT

Explanatory note: 114 pp., 1 table, 58 figures, 3 appendices, 34 sources.

MES-SYSTEM, PRODUCTION, OPTIMIZATION, SOFTWARE, TELEPHONE BATTERIES,

The object of development is the production process of telephone batteries.

The subject of development is an automated system for designing and optimizing production processes.

The purpose of the work is to develop a component of the MES system aimed at optimizing the production processes of the enterprise for the production of telephone batteries.

Research methods – review of enterprise requirements for a specific system, modeling of production processes using UML diagrams, creation of a MySQL database, development of the server and client part of the application using the Java language and auxiliary frameworks.

In the qualification work, topical issues on the topic were considered, solutions for optimizing processes in modern production were proposed. The software of the MES system component was designed and developed for three employees of the enterprise, and its functionality was checked.

The developed software can be integrated into production to ensure its integrity, minimize equipment downtime and improve the efficiency of the enterprise as a whole.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД – База Даних;

СУБД – Система Управління Базами Даних;

MES – Manufacturing Execution System;

MTBF – Mean Time Between Failures;

MTTA – Mean Time To Acknowledge;

MTTR – Mean Time To Repair;

OEE – Overall Equipment Effectiveness;

ORM – Object-Relational Mapping;

UML – Unified Modeling Language;

.

ЗМІСТ

Вступ 12

1 Аналіз предметної області 14

1.1 Концепція MES-системи та її роль в оптимізації виробничих процесів підприємства. 14

1.2 Огляд основних компонентів системи MES 15

1.3 Актуальність та переваги впровадження системи на сучасних підприємствах 17

1.4 Аналіз проблем автоматизації та оптимізації робочих процесів на виробництві телефонних акумуляторів за умови відсутності централізованої системи управління 18

1.5 Постановка задачі 21

1.5.1 Структура створюваної системи 21

1.5.2 Основні критерії, за якими MES-система оптимізує процеси підприємства 23

2 Опис вимог до розроблюваного MES-системи 28

2.1 Визначення системних вимог для функціонування системи 28

2.2 Використання мови моделювання UML для проєктування роботи системи . 28

2.2.1 Розробка діаграми варіантів використання для компоненту MES-системи 29

2.2.2 Розробка діаграм взаємодії для компоненту MES-системи 33

2.2.3 Розробка діаграми класів для компоненту MES-системи 36

2.3 Опис основних процесів виробництва телефонних акумуляторів та графіку його роботи 37

2.4 Опис розрахунків даних для імітаційного моделювання системою процесів виробництва 39

2.4.1 Розрахунки для міксера з вимішування активного матеріалу (суспензії) 40

2.4.2 Розрахунки для міксера з вимішування електроліту 42

2.4.3 Розрахунки для машини з нанесення пасти на електроди 44

2.4.4 Розрахунки для сушильної печі 45

2.4.5 Розрахунки для машини, яка проводить каландрування 45

2.4.6 Розрахунки для машини, яка проводить розділення електродів 46

2.4.7 Розрахунки для машини, яка вкладає частини акумулятора у герметичний блок 47

2.4.8 Розрахунки для вакуумної машини з заливки електроліту 47

2.4.9 Розрахунки для вакуумної машини з герметизації блоку акумулятора 48

2.5 Охорона праці 48

3 Створення бази даних предметної області 51

3.1 Поняття ER-моделювання 51

3.2 Створення таблиць бази даних 52

3.2.1 Опис зв’язків 53

3.2.2 Опис таблиці mixerpasta 54

3.2.3 Опис таблиці mixerelectrolyte 55

3.2.4 Опис таблиці syringemachines 56

3.2.5 Опис таблиці dryoven 56

3.2.6 Опис таблиці hydrpress 57

3.2.7 Опис таблиці lasercutter 58
3.2.8 Опис таблиці stackmash 59
3.2.9 Опис таблиці cellelectrolytem 60
3.2.10 Опис таблиці sealingchamber 61
3.2.11 Опис таблиці machines 62
3.2.12 Опис таблиці repair 63
3.2.13 Опис таблиці work_program 65
3.2.14 Опис таблиці users 66
3.3 Нормалізація таблиць баз даних 67
4 Опис прийнятих проєктних рішень 72
4.1 Вибір мови програмування та допоміжного фреймворку 72
4.2 Вибір СУБД та фреймворку для взаємодії з нею 75
4.3 Вибір середовища програмної реалізації та додатків для взаємодії з базою даних та локальним сервером 77
4.4 Налаштування системи авторизації користувачів 78
4.5 Розробка серверної частини системи 83
4.5.1 Контролер ManufacController 84
4.5.2 Код контролеру ProgramController 84
4.5.3 Код контролеру ReportsController 86
4.5.4 Код контролеру DispProcessesController 88
4.5.5 Код контролеру MonitoringController 88
4.5.6 Код контролеру ReplaceMachinesController 89
4.5.7 Код контролеру DailyReportController 90

4.5.8 Код репозиторію MixerpastaRepository 90
4.5.9 Код репозиторію WorkProgramRepository 91
4.5.10 Розробка тригерів 91
4.6 Розробка клієнтської частини системи 92
4.6.1 HTML-файл header 93
4.6.2 HTML-файл work-prog 94
4.6.3 HTML-файл base-plan 97
4.6.4 HTML-файл reports 103
4.6.5 HTML-файл monitoring 104
4.6.6 HTML-файл replace-mach 107
Висновки 109
Перелік джерел посилання 110
Додаток А Керівництво користувача 115
Додаток Б Текст програми 138

ВСТУП

На фоні стрімкого розвитку виробничих технологій в останні десятиліття автоматизація стає все більш актуальною для підприємств у всіх галузях. Зростаюча конкуренція та потреба в ефективному розподілі ресурсів примушують сучасні підприємства шукати нові шляхи оптимізації своєї виробничої діяльності. В цьому контексті системи виробничого управління (Manufacturing Execution System, MES) стають ключовим інструментом для підвищення продуктивності, якості та конкурентоспроможності підприємств. Детальне вивчення та аналіз вимог виробничих підприємств, розробка та імплементація ефективного компонента MES-системи мають на меті підвищення ефективності управління виробництвом, зниження витрат та виробничих простоїв, а також поліпшення якості виготовленої продукції.

Мета роботи – розробка компонента MES-системи, спрямованого на оптимізацію виробничих процесів підприємства з виробництва телефонних акумуляторів.

Об’єкт розробки – процес виробництва телефонних акумуляторів.

Предмет розробки – автоматизована система для проєктування та оптимізації процесів виробництва.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області;

– виділити основні вимоги до розробки компоненту MES-системи

– розробити базу даних предметної області;

– провести впровадження системи у виробництво;

– оформити кваліфікаційну роботу згідно ДСТУ 3008:2015 [1], а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151

«Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Системна інженерія» [2] і з дипломним проєктуванням для студентів усіх форм навчання [3].

За результатами роботи було опубліковано тези доповіді у збірнику університету [4] та наукову статтю у науково-технічному журналі [5].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Концепція MES-системи та її роль в оптимізації виробничих процесів підприємства.

MES-система являє собою програмне забезпечення, яке відіграє ключову роль в управлінні виробничими процесами на підприємстві. Вона забезпечує інтеграцію між рівнями виробництва (рівень автоматизації, рівень управління та рівень корпоративного управління), а також дозволяє збирати, аналізувати та використовувати дані про виробничі операції для подальшого прийняття управлінських рішень [6].

Роль MES-системи в оптимізації виробничих процесів є критично важливою для сучасних підприємств, особливо в умовах ринку, що постійно змінюється, і конкуренції, що збільшується. Нижче наведені деякі ключові аспекти, які підкреслюють значення MES-системи оптимізації виробничих процесів:

– автоматизація та стандартизація операцій: MES-система дозволяє автоматизувати виробничі процеси, що зменшує ймовірність помилок та підвищує якість продукції, що випускається. Це також дозволяє скоротити час виконання операцій та покращити продуктивність праці;

– управління ресурсами: MES-система допомагає оптимізувати використання ресурсів підприємства, таких як обладнання, матеріали, трудові ресурси та енергія. Вона дозволяє ефективно планувати та розподіляти ресурси відповідно до виробничих потреб, мінімізуючи простої та створення надлишкових запасів;

– моніторинг та контроль виробничих процесів: MES-система забезпечує безперервне відстеження правильності виконання виробничих операцій та надає інформацію про поточний стан процесів в реальному часі. Це дозволяє оперативно

реагувати на відхилення від норми, запобігати простоям обладнання та мінімізувати втрати продукції;

– оптимізація виробничого планування: MES-система допомагає оптимізувати планування, одночасно враховуючи кількість ресурсів, терміни виконання замовлень та зміни у виробничих умовах. Вона дозволяє створювати ефективніші плани, зменшуючи час простою обладнання та скорочуючи фінансові витрати на виробництво;

– управління якістю: MES-система надає функціонал для забезпечення контролю якості на всіх етапах виробництва, від прийому сировини до фінального формування продукції. Вона допомагає виявляти та усувати дефекти на ранніх етапах, що дозволяє скоротити кількість браку.

1.2 Огляд основних компонентів системи MES

Компоненти MES-систем являють собою різні модулі та функціональні блоки, призначені для управління та оптимізації різних аспектів виробничих процесів на підприємстві. На рисунку 1.1 наведені основні компоненти систем.

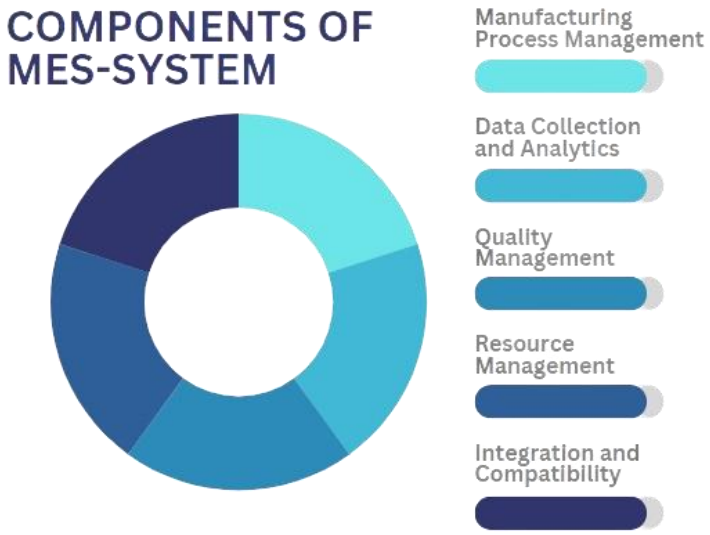

Рисунок 1.1 – Компоненти MES-систем

Детальний опис наведено нижче:

– управління виробничим процесом (Manufacturing Process Management): модуль, який керує виконанням виробничих операцій, відстеженням статусів замовлень, операцій, обладнання та матеріалів. Він включає в себе функціональні можливості для створення та управління робочими замовленнями, складання розкладів виробництва, моніторингу виконання операцій в реальному часі та оптимізації виробничих потоків.

– збір даних та аналітика (Data Collection and Analytics): модуль для збору, аналізу та візуалізації даних про виробничі процеси. Він включає в себе можливості для автоматичного збору даних про час виконання операцій, використання обладнання, витрати матеріалів, а також аналізу цих даних для виявлення вузьких місць, оптимізації виробничих процесів і прийняття управлінських рішень.

– управління якістю (Quality Management): модуль для контролю якості продукції всіх етапах виробництва. Він включає в себе функціонал для проведення перевірок, контролю відповідності продукції стандартам якості, управління браком та відхиленнями від норми, а також аналізу даних про якість для пошуку причин дефектів та поліпшення виробничих процесів.

– управління ресурсами (Resource Management): модуль для оптимізації використання ресурсів підприємства, таких як обладнання, матеріали, трудові ресурси та енергія. Він включає в себе функціонал для планування та розподілу ресурсів, обліку та моніторингу їх використання, оптимізації витрат і скорочення втрат.

– інтеграція та сумісність (Integration and Compatibility): модуль для інтеграції MES-систем з іншими інформаційними системами підприємства, такими як планування ресурсів підприємства (ERP), управління виробництвом (SCADA), системи управління якістю і т.д. Забезпечує обмін даними між різними системами, забезпечуючи цілісність та достовірність даних та ефективну взаємодію між різними рівнями та функціональними областями підприємства.

1.3 Актуальність та переваги впровадження системи на сучасних підприємствах

Одним з найбільш вдалих прикладів впровадження системи у виробничі процеси є компанія Littelfuse Semiconductor, заснована в 1927 році Littelfuse зі штаб-квартирою в Чикаго, штат Іллінойс, Сполучені Штати. Вона є виробничою компанією, що займається виробництвом промислових технологій у понад 15 країнах [7].

Зіткнувшись із питанням впровадження рішення Industry 4.0 для нового заводу з високоавтоматизованими виробничими лініями для створення силових напівпровідникових компонентів (завод виробляє силові модулі, високотемпературні дискрети з низькими втратами, тиристори середньої та великої потужності), Littelfuse вирішила співпрацювати з компанією Cantier, яка є постачальником галузевої системи управління виробництвом на базі штучного інтелекту, щоб розробити сучасне рішення для виробництва, яке можна конфігурувати та масштабувати відповідно до поточних і майбутніх потреб.

Компанія Littelfuse реалізувала рішення Cantier MES 4.0 у 4 секціях на понад 50 машинах, які потрібно було підключити для збору даних у реальному часі. Рішення Cantier MES 4.0 було створено ексклюзивно для Littelfuse-Semiconductors і забезпечило наступні можливості для їх виробничих операцій: з’єднання в режимі реального часу між усіма машинами (на основі ПЛК і з підтримкою SEC/GEM) через шлюз IIoT для впровадження Cantier MES 4.0; відстеження матеріалів/інструментів у реальному часі; керування технологіями виробництва; трекінг та трейсинг на рівні партії/перевізника/підрозділу; система прогнозування і профілактичного обслуговування; моніторинг журналу тривог і подій; детальні інформаційні панелі OEE, SPC і технічного обслуговування обладнання; інтеграція з ERP системами; виробничий штучний інтелект.

Впровадження Cantier MES 4.0 надало Littelfuse можливість не тільки досягти

загального підвищення продуктивності, а й реалізувати швидке та стабільне покращення MTTR, MTBF та OEE.

1.4 Аналіз проблем автоматизації та оптимізації робочих процесів на виробництві телефонних акумуляторів за умови відсутності централізованої системи управління

Компанії, що не мають злагодженого підходу до управління, стикаються з рядом проблем, що ускладнюють їх робочі процеси та впливають на ефективність виробництва. Однією з таких проблем є неструктурованість та роз'єднаність даних про цикли виробництва, що ускладнює моніторинг робочих процесів, відстеження стану обладнання та матеріальних ресурсів, а також координацію роботи персоналу.

Ще однією проблемою є недостатня автоматизація процесів. Без централізованої системи управління більшість операцій виконуються вручну або за допомогою окремих ізольованих програм, що призводить до збільшення часу на виконання завдань та ймовірності помилок. Недоцільне використання робочого часу та ресурсів може суттєво вплинути на продуктивність підприємства та його конкурентоспроможність на ринку.

Також актуальним є питання того, що підприємство має обмежений доступ до інформації про свої виробничі процеси. Без централізованої системи збору та аналізу даних складно виявляти проблемні аспекти виробничого процесу та приймати обґрунтовані управлінські рішення для їх вирішення.

Отже, відсутність централізованої системи управління може призвести до низки проблем, таких як неефективне управління даними та ресурсами, неефективна автоматизація процесів та втрата конкурентоспроможності підприємства.

Зважаючи на вищевказані вади і недоліки виробництва, а також на функціональні можливості системи, описані у пунктах 1.1 – 1.2 є доцільним впровадження системи управління MES (Manufacturing Execution System) у виробництво телефонних акумуляторів. На рис. 1.2 представлені деякі переваги, які мають підприємства з виробництва акумуляторів при використанні MES-системи в порівнянні з тими, які нею не користуються:

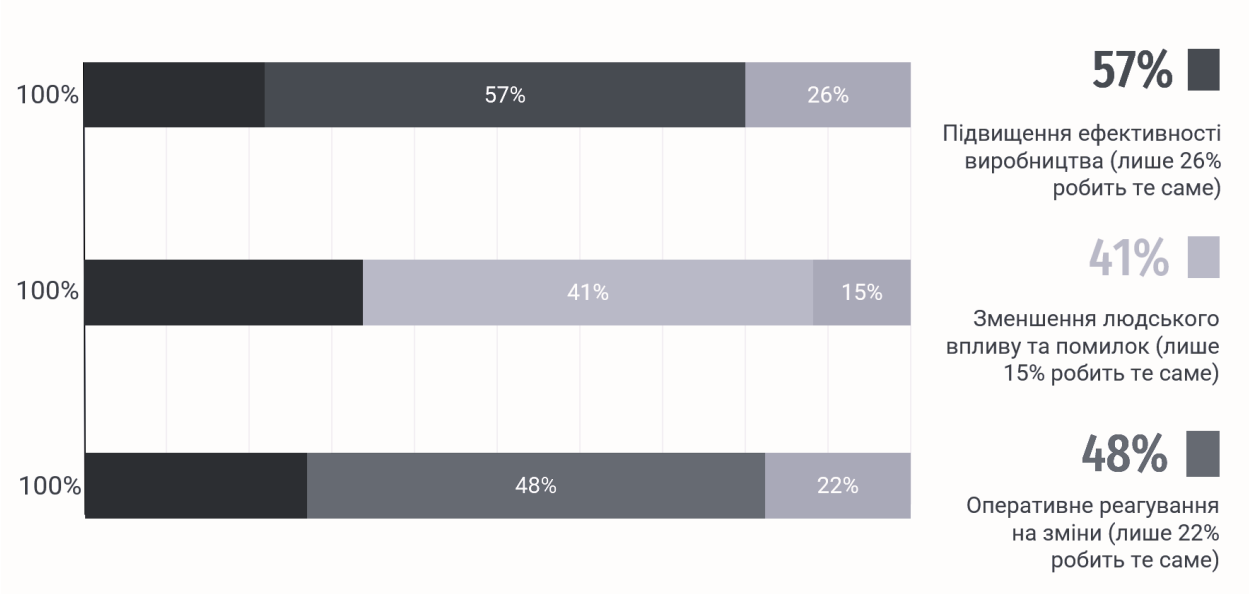

Рисунок 1.2 – Основні операційні переваги фірм, які інтегрують MES-системи [4]

Для впровадження системи у виробництво необхідно виділити такі етапи:

– аналіз потреб підприємства: перший крок передбачає докладне вивчення потреб та вимог підприємства, включаючи аналіз існуючих проблем виробництва та областей, які потребують оптимізації;

– визначення вимог для системи: на цьому етапі проєктується MES-система, яка найкращим чином відповідає потребам підприємства. Враховуються функціональні можливості системи, її сумісність з існуючим обладнанням та програмним забезпеченням;

– розробка та налаштування: після висування вимог до системи проводиться її розробка та налаштування під конкретні потреби підприємства. Цей етап включає в себе налагодження функціональності системи з урахуванням специфіки виробничих процесів;

– впровадження та навчання персоналу: MES-система впроваджується на підприємство, а персонал проходить інструктаж з її використання. Це включає навчання з використання програмного забезпечення, роботи з даними та аналізу інформації, яка надається системою;

– підтримка та моніторинг: після впровадження необхідна постійна підтримка та відслідковування недоліків роботи системи. Це включає в себе вирішення будь-яких збоїв та неточностей, оновлення програмного забезпечення та надання консультативної підтримки персоналу з використання оновленого функціоналу системи.

Впровадженням MES на виробництві буде досягнуто таких результатів:

– підвищено ефективність виробництва;

– підвищено якість продукції;

– підвищено рівень автоматизації;

– збільшено рівень гнучкості виробничих процесів;

– підвищено рівень точності прийняття управлінських рішень;

– покращено рівень співпраці та комунікації між різними відділами виробництва;

– зменшено час виробництва продукції та прискорено виконання замовлень клієнтів.

1.5 Постановка задачі

Головною метою кваліфікаційної роботи є розробка компоненту MES-системи Управління виробничим процесом для оптимізації виробництва телефонних акумуляторів для моделі телефону . Для злагодженої роботи системи та її правильної інтеграції необхідно дотримуватися етапів впровадження, описаних у пункті 1.4.

1.5.1 Структура створюваної системи

Система поєднує в собі інтерфейс для трьох користувачів: проєктувальника підприємства, який за допомогою імітаційного моделювання процесів виробництва складає робочі програми для кожної машини з урахуванням коефіцієнту її максимальної завантаженості та термінів виконання замовлення від клієнта, диспетчера, який відслідковує правильність виконання робочих процесів та відхилень від норми шляхом моніторингу показників кожної машини, які надаються йому системою в реальному часі. Він також має можливість створювати рапорти для виконання ремонту ремонтною бригадою. Третім користувачем системи є бригадир ремонтної бригади, який отримує рапорт на ремонт від диспетчера з усіма необхідними деталями, проводить разом з бригадою відновлення пошкоджень та доповнює рапорт деталями. Також при потребі заміни серйозно пошкодженої машини на іншу зі складу бригадир має інтерфейс для швидкого встановлення необхідної робочої машини всередині бази даних для подальшого динамічного відображення оновлень у інших користувачів.

Розроблена система реалізує наступний функціонал:

– можливість отримання проєктувальником паперової або електронної копії таблиці з інформацією про замовлення від клієнта;

– можливість створення робочих програм проєктувальником;

– імітація за допомогою графічного відображення процесів виготовлення частин акумулятора машинами;

– можливість редагування необхідних параметрів робочих програм, що потім динамічно відображається на графіку імітаційного моделювання;

– автоматичне складання програми виготовлення одного замовлення, яка включає в себе всі робочі програми машин з оновлюваними в реальному часі даними;

– можливість отримання паперової або електронної копії програми виготовлення одного замовлення і персонального завдання для кожної машини, які можуть отримувати як проєктувальник, так і диспетчер;

– передавання параметрів машин на дисплей диспетчеру в реальному часі для відслідковування недоліків та неточностей у роботі підприємства;

– можливість керування процесами виробництва диспетчером та їх віддаленим запуском;

– система сповіщень, яка повідомляє диспетчера про проблеми або завершення того чи іншого процесу.

– можливість створення рапортів на ремонт пошкоджень або перевірку аномалій у роботі машин;

– можливість генерації звіту з виконаними за поточний день програмами та ремонтними роботами, доступ до якого має диспетчер і проєктувальник;

– можливість відображення створених диспетчером рапортів у інтерфейсі бригадира ремонтної бригади з можливістю їх редагування та внесення додаткової інформації;

– інтерфейс для бригадира, який надає можливість динамічно замінювати машини у базі даних, якщо вони були пошкоджені та відправлені на склад.

1.5.2 Основні критерії, за якими MES-система оптимізує процеси підприємства

Для оптимізації процесів підприємства шляхом введення в нього MES-системи було виділено основні критерії оптимізації, які необхідно враховувати про розробці компоненту Управління робочим процесом:

– коефіцієнт завантаженості обладнання: це показник, який відображає рівень використання виробничих потужностей або обладнання за визначений період часу. Для його розрахунку використовується формула:

$$K_{\text{з}} = \frac{C_{pi}}{C_{\text{П}pi}} \times 100\%, \tag{1.1}$$

де $K_{\text{з}}$ – коефіцієнт завантаженості обладнання;

$C_{\text{рі}}$ – розрахункова кількість обладнання на даній операції;

$C_{\text{прі}}$ – прийнята кількість обладнання на даній операції.

Потім визначається середній коефіцієнт завантаження лінії:

$$K_{\text{зСР}} = \frac{\sum \text{С}_{pi}}{\sum \text{С}_{\text{П}pi}} \times 100\%. \tag{1.2}$$

Коефіцієнт завантаженості $K_{\text{з}}$ при поточно-масовому виробництві для кожного типу обладнання повинен бути 0,75, або 75%. У середньому по цеху $K_{\text{зСР}}$ повинен бути 80%.

Способи приведення коефіцієнту завантаження $K_{\text{з}}$ до необхідного рівня за допомогою MES-системи:

– покращення планування виробничих процесів, їх імітація перед введенням робочих програм у підприємство для унеможливлення простоїв та вищої за норму завантаженості обладнання;

– регулярне технічне обслуговування та вчасний ремонт для уникнення непередбачених простоїв.

Впровадження розрахунку коефіцієнту завантаженості $K_з$ в робочу практику надає можливість максимально використовувати виробничі потужності та більш збалансовано користуватися обладнанням, що значно подовжує термін його працездатності та знижує вірогідність поломок.

– час виконання замовлення: показник, який відображає всі етапи від отримання замовлення до його доставки клієнту. Для його розрахунку використовується формула:

$$T_{заг} = T_{зам} + T_{вир} + T_{дост}, \quad (1.3)$$

де $T_{заг}$ – загальний час виконання замовлення;

$T_{зам}$ – час обробки замовлення;

$T_{вир}$ – час виготовлення (включає в себе підготовку, налагодження та саме виробництво);

$T_{дост}$ – час доставки готової продукції клієнту.

Шляхи зменшення часу виконання замовлення $T_{заг}$, які пропонує впровадження MES-системи:

– розподілення робочих задач таким чином, щоб вмістити максимальну кількість замовлень в невеликий проміжок часу без шкоди для обладнання;

– система моніторингу стану машин та процесів виконання робочих задач для швидкого реагування робочим персоналом на аномалії та їх подальше усунення;

– оперативне усунення проблем виробництва та фіксація точного часу, витраченого на ремонт, для подальшого планування робочих задач у врахуванням даного параметру.

Час виконання замовлення $T_{заг}$ є важливим критерієм, адже він безпосередньо впливає на рівень задоволеності клієнта сервісом даної компанії, а також рівень конкурентноспроможності підприємства.

– показник MTBF: використовується для оцінки надійності системи. Він являє собою середній час між відмовами обладнання або системи. Ціллю для більшості компаній є збереження даного показнику на якомога вищому рівні, тому що чим більше часу проходить між поломками – тим надійніше система. MTBF розраховується з використанням середнього арифметичного.

Необхідно взяти дані за визначений період та розділити загальний час роботи за цей період на кількість збоїв.

$$MTBF = \frac{T_{заг}}{N}, \tag{1.4}$$

де $T_{заг}$ – загальний час роботи обладнання в годинах;

N – кількість відмов обладнання за цей період.

Шляхи підвищення значення показнику MTBF, які пропонує впровадження MES-системи:

– моніторинг процесів виробництва в реальному часі для швидкого реагування на аномалії та їх усунення для унеможливлення повної відмови машини внаслідок несвоєчасного ремонту;

– генерація системою звітів, які дозволяють аналізувати причини відмов та розробляти ефективні мірі по їх запобіганню в подальшому.

Показник MTBF є дуже важливим для роботи підприємства, адже він надає можливість оцінити продуктивність та стабільність роботи обладнання. Даний показник часто використовують при роботі з клієнтами для демонстрації рівня надійності компанії.

– показник MTTR (Mean Time To Repair): означає середній час, який необхідно витратити на ремонт обладнання. Він включає в себе як час ремонту, так

і подальше тестування працездатності машини. Дана метрика враховує весь час до того моменту, коли обладнання не відновить роботу повністю. MTTR розраховується за допомогою суми загального часу, витраченого на ремонт, протягом заданого періоду, діленої на кількість ремонтів.

$$MTTR = \frac{T_{\text{рем}}}{N}, \tag{1.5}$$

де $T_{\text{рем}}$ – загальний час у годинах, витрачений на ремонт протягом визначеного періоду;

Шляхи зниження значення показнику MTTR, які пропонує впровадження MES-системи:

– швидке реагування ремонтною бригадою на проблеми обладнання за рахунок відстеження процесів у реальному часі, що зменшує час виконання ремонту та критичність поломки;

– точна фіксація часу, витраченого на ремонт, для аналізу та подальшого прийняття рішень з метою підвищення ефективності роботи бригади.

Використання вимірювання показнику MTTR надає можливість підприємству оцінювати якість та ефективність обслуговування машин ремонтною бригадою, а також підтримувати їх на потрібному рівні.

– показник MTTA: означає середній час, необхідний для підтвердження отримання повідомлення про несправність або відмову обладнання. Він використовується для оцінки оперативності реагування на інциденти. Розраховується показник так:

$$MTTA = \frac{T_{\text{заг}}}{N}, \tag{1.6}$$

де $T_{заг}$ – загальний час, витрачений на підтвердження всіх інцидентів протягом визначеного періоду;

За рахунок централізованості MES-системи і відсутності додаткових підсистем, взаємодія з якими сповільнила б передачу повідомлення, при занесенні інформації про проблему від одного користувача ці дані одразу з’являються у іншого. Це значно пришвидшує процесс отримання інформації про проблеми та залишає серед можливих причин підвищення показника лише людський фактор.

Використання вимірювання показнику MTTA надає можливість підприємству оцінювати оперативність реакції персоналу на зміни а також якість і ефективність їх роботи.

2 ОПИС ВИМОГ ДО РОЗРОБЛЮВАНОГО КОМПОНЕНТУ MES-СИСТЕМИ

2.1 Визначення системних вимог для функціонування системи

Для злагодженої роботи системи необхідно окреслити ряд основних системних вимог, які повинні виконуватися при розробці компоненту:

– система повинна бути реалізована за допомогою клієнт-серверної архітектури (архітектура, яка передбачає наявність клієнтського застосунку, підключеного до сервера, який пов’язаний з базою даних);

– сервер повинен бути реалізований за допомогою системи управління базами даних MySQL;

– клієнтська частина системи має бути реалізована у вигляді вебсайту з різними вкладками, інформація на якому відтворюється за допомогою мов розмітки та стилів.

2.2 Використання мови моделювання UML для проєктування роботи системи

UML – уніфікована мова моделювання, яка використовується у парадигмі об'єктно-орієнтованого програмування та є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проєктування й документування програмних систем. Вона не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація [8].

UML надає набір графічних символів та правил, які дозволяють створювати абстрактні моделі різних аспектів системи, таких як її структура, поведінка, взаємодія між компонентами та навіть процес її розробки.

Основні елементи UML включають:

– діаграми класів (Class diagrams): показують структуру системи, описуючи класи, їх атрибути та методи, а також зв'язки між класами.

– діаграми взаємодії (Interaction diagrams): є групою діаграм, які моделюють взаємодію між об'єктами або компонентами в системі. Вони включають в себе діаграми послідовностей (Sequence Diagrams) та діаграми комунікації (Communication Diagrams);

– діаграми станів (State diagrams): моделюють різні стани, у яких може бути об'єкт чи система, і переходи між цими станами.

– діаграми варіантів використання (Use Case diagrams): описують функціональні вимоги системи, показуючи, як вона взаємодіятиме з персоналом та іншими системами в різних сценаріях використання.

– діаграми компонентів (Component diagrams): показують структуру фізичної реалізації системи, включаючи компоненти та їх взаємозв'язки, а також розгортання цих компонентів на апаратному устаткуванні.

Для проєктування роботи компоненту MES-системи Управління виробничим процесом на виробництві телефонних акумуляторів буде задіяно діаграми варіантів використання, класів та взаємодії. Для створення діаграм було обрано додаток Star UML.

2.2.1 Розробка діаграми варіантів використання для компоненту MES-системи

Діаграма варіантів використання – це тип поведінкової діаграми UML, який часто використовується для аналізу різних систем. Вони дозволяють візуалізувати різні типи ролей та те, як ці ролі взаємодіють із системою. На діаграмах варіантів

використання відображається взаємодія між прецендентами, що представляють функції системи, та дійовими особами (акторами), які представляють людей або системи, які одержують або передають інформацію до цієї системи. Розроблена діаграма варіантів використання представлена на рис. 2.1.

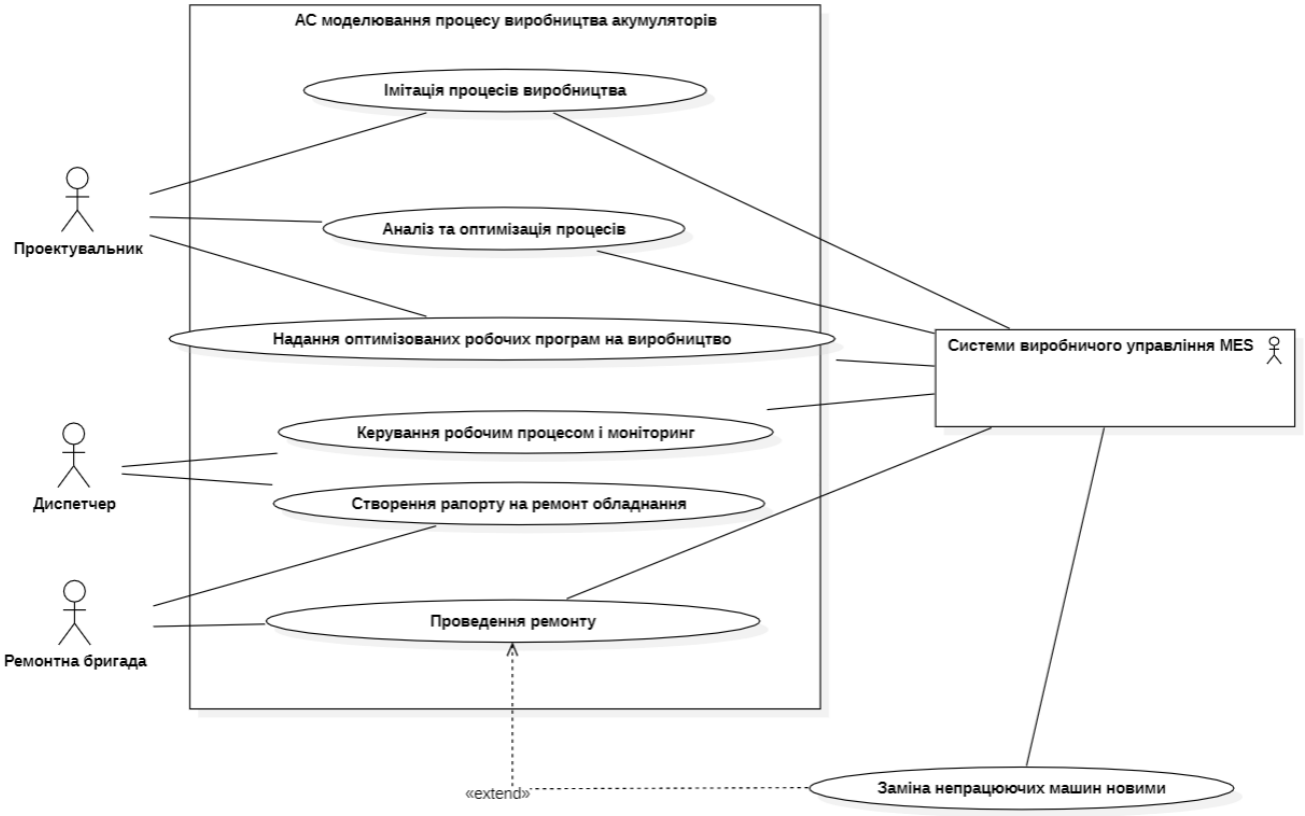

Рисунок 2.1 – Розроблена діаграма для компоненту MES-системи

При реалізації діаграми для розробки компоненту було виділено таких акторів:

– проєктувальник: фахівець, який відповідає за імітацію процесів виробництва, їх подальшу оптимізацію та створення робочих програм, концепцій та технічних специфікацій проєктів;

– диспетчер: відповідає за керування робочим процесом, моніторинг злагодженості роботи виробництва та подання звітності про стан підприємства;

– ремонтна бригада: відповідає за ремонт пошкоджених машин та їх елементів, а також за їх заміну, у разі потреби, на нові;

– система виробничого управління MES: бере участь у автоматизації процесів виробництва, їх оптимізації та моделюванні.

Було виділено наступні прецеденти:

– Імітація процесів виробництва: прецендент дозволяє проєктувальнику вносити необхідні параметри для моделювання різних процесів виробництва.

Основний успішний сценарій:

а) проєктувальник отримує дані про замовлення клієнта з системи;

б) проєктувальник створює робочі програми для кожної машини підприємства, вказуючи дані з замовлення;

в) на основі створених програм система моделює у графічному вигляді тривалість процесів.

– Аналіз та оптимізація процесів: прецендент дозволяє проєктувальнику аналізувати результат моделювання та вносити правки.

Основний успішний сценарій:

а) проєктувальник отримує від системи результат моделювання за початковими параметрами. Процеси, які перевищують коефіцієнт завантаженості машини, відображені іншим кольором;

б) проєктувальник змінює необхідні вхідні параметри для ліквідації перезавантажень та підганяє їх під очікувану клієнтом дату виконання замовлення;

в) система реагує на зміни та демонструє оновлений приклад моделювання системи.

– Надання оптимізованих робочих програм на виробництво: прецендент дозволяє проєктувальнику надавати створені програми на виготовлення.

Основний успішний сценарій:

а) система формулює оновлені робочі програми для машин в різні блоки для кожного замовлення;

б) система створює персональні завдання для кожної машини на основі робочих програм;

в) проєктувальник надсилає створені персональні завдання та робочі програми на виробництво.

– Керування робочим процесом та моніторинг: дозволяє диспетчеру запускати, контролювати та припиняти виробництво. Він відслідковує незмінність характеристик процесу виробництва та забезпечує правильність його виконання.

Основний успішний сценарій:

а) диспетчер бачить заплановані на сьогодні робочі програми та аналізує персональні задачі для кожної машини;

б) диспетчер запускає виробництво та відслідковує показники машин;

в) при виконанні кожної робочої програми система інформує диспетчера за допомогою системи сповіщень;

г) після успішного завершення усіх завдань на поточний день система формує звіт про виконані програми, який можуть переглядати інші працівники виробництва та керівництво.

– Створення рапорту на ремонт обладнання: дозволяє диспетчеру створювати рапорти на ремонт в разі відхилень показників машин від норми. Дозволяє бригадиру ремонтної бригади вносити в рапорт деталі ремонту.

Основний успішний сценарій:

а) диспетчер фіксує неполадки у одній із робочих машин;

б) диспетчер зупиняє робочі процеси;

в) диспетчер створює рапорт із деталями поломки та направляє його бригадиру ремонтної бригади;

г) диспетчер очікує на завершення ремонту та відредагований бригадиром рапорт, а після робить перезапуск робочих процесів.

– Проведення ремонту: дозволяє ремонтній бригаді провести ремонт на проблемній ділянці виробництва

Основний успішний сценарій:

а) бригадир ремонтної бригади отримує рапорт від диспетчера та готує ремонтну бригаду і необхідні інструменти для виконання ремонту;

б) бригада прибуває на місце, оглядає поломку;

в) бригада робить висновок: якщо машина підлягає відновленню, то ремонтується пошкодження та машина запускається у тестовому режимі. При успішному проходженні перевірки бригадир заповнює рапорт деталями, вказує точний час, який було витрачено на ремонт. Якщо машина не підлягає відновленню, то виконується розширення Заміна непрацюючих машин новими:

а) бригада робить висновок, що машина серйозно пошкоджена та підлягає заміні;

б) бригада відключає машину і проводить транспортування та підключення запасної машини зі складу;

в) бригада запускає машину у тестовому режимі. При успішному проходженні перевірки бригадир заповнює рапорт деталями, вказує точний час, який було витрачено на ремонт та робить заміну працюючої моделі машини на іншу у додатку.

2.2.2 Розробка діаграм взаємодії для компоненту MES-системи

Діаграми взаємодії в UML, як було вказано вище, включають два основні типи: діаграми послідовностей та діаграми комунікації. Обидва типи діаграм призначені для візуалізації взаємодії між об'єктами чи компонентами у межах системи.

Діаграми послідовностей показують послідовність повідомлень, що передаються між об'єктами у часі. Об’єкти в них мають так звані лінії життя, які зображуються вертикальними пунктирними лініями, розташованим під об’єктом. Вздовж ліній розташовані блоки (фокуси управління), які є показником виконання

об’єктом деякої операції. Повідомлення між об’єктами мають вигляд горизонтальних ліній, що направлені між фокусами управління.

Діаграми комунікації UML відображають процес передачі повідомлень всередині системи. Вони ґрунтуються на взаємовідносинах у системі та на потоці повідомлень, але не відображають порядок, час чи послідовність їх передачі. Об'єкти в діаграмі зображені у вигляді прямокутників. Взаємодія між об'єктами відображається стрілками, що вказують напрямок повідомлень.

При створенні діаграм взаємодії для компоненту MES-системи було обрано прецендент Керування робочим процесом та моніторинг. Створені діаграми реалізують процес керування виробничим процесом з боку диспетчера. На рисунку 2.2 зображена діаграма послідовності системи.

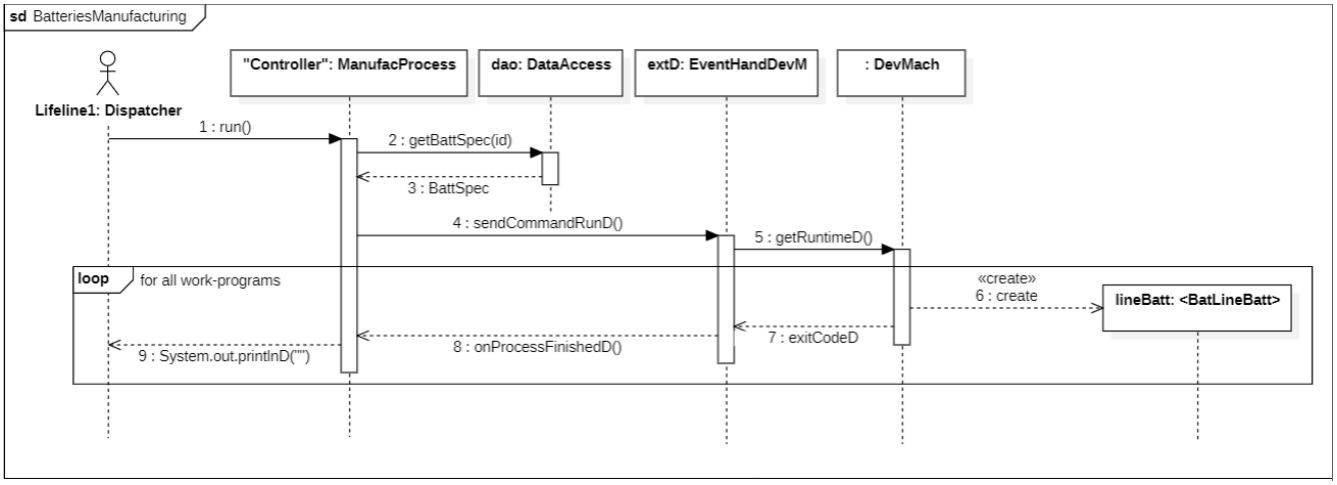

Рисунок 2.2 – Створена діаграма послідовності для проєктованої системи

Діаграма має головний елемент Controller, який називається ManufacProcess. За його допомогою команди диспетчера розподіляються у правильні класи. Він також відповідає за інформування диспетчера про завершення того чи іншого процесу. Для отримання робочої програми контроллер звертається до бази даних DataAccess перед подачею команди на старт виробництва. Для взаємодії з фізичними машинами на підприємстві до них підключені обробники подій, які

передають їм команди від контроллера на початок роботи та приймають від машини інформацію про те, що завдання виконано. Також на діаграмі присутній фрейм loop, який показує, що дана операція на виробництві повторюється визначену кількість разів.

На рисунку 2.3 зображена діаграма комунікацій для системи.

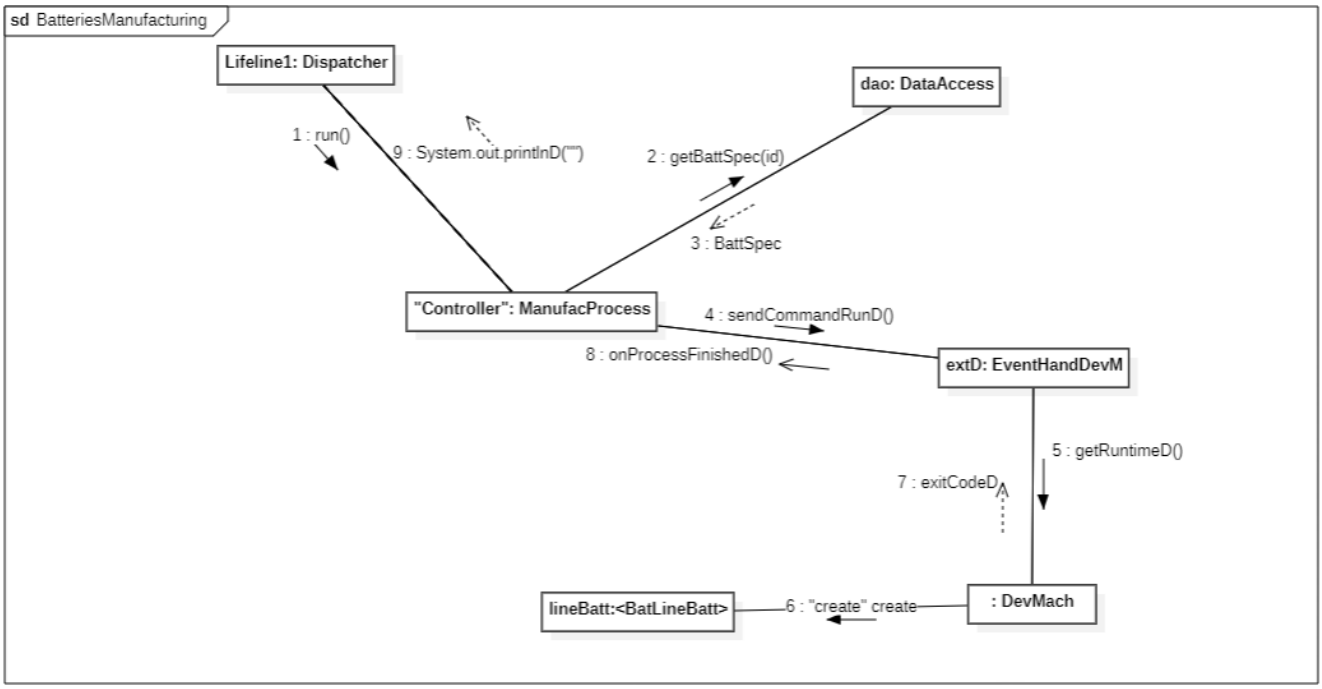

Рисунок 2.3 – Створена діаграма комунікацій для системи

На діаграмах наведені деякі методи, роль яких пояснено нижче:

– run(): запуск контролера і всіх процесів;

– getBattSpec(id): запит до бази даних для отримання інформації щодо партії;

– BattSpec: отримання даних про партію акумуляторів з бази даних;

– sendCommandRunD(): надсилання команди від контролера про запуск машин до обробника подій машин;

– getRuntimeD(): команда від обробника до машин, яка наказує почати роботу. Машини створюють акумулятори і вони заносяться в список lineBatt:<BatLineBatt>;

– exitCode: повернення від машин до обробника числового значення, що свідчить про завершення роботи;

– onProcessFinished(): обробник повідомляє контролер про завершення роботи машин;

– System.out.println(“”): повідомлення контролером диспетчера про завершення операції через виведення відповідного тексту;

2.2.3 Розробка діаграми класів для компоненту MES-системи

Діаграма класів описує типи об'єктів системи та різного роду статичні відносини, які існують між ними. На діаграмах класів відображаються властивості класів, операції і обмеження, які накладаються на зв'язки між об'єктами. На діаграмі клас зображується у вигляді прямокутника, розділеного на три частини: ім'я класу, його атрибути та його операції. Відношення між класами наводиться у вигляді різних типів зв’язку:

– залежність: зв’язок, який вказує на те, що зміна незалежного елемента обов’язково призведе до зміни залежного. Вона представлена у вигляді стрілки з пунктирною лінією;

– асоціація: показує, що об’єкти одного класу можуть переміщуватися до іншого і навпаки (якщо асоціація множинна). Вона представлена стрілкою, якщо це односторонній зв’язок, або лінією, якщо множинний;

– агрегація: використовується у випадку, коли клас є контейнером або колекцією інших. Якщо контейнер буде знищено, то його зміст залишиться. Відображається зв’язок у вигляді лінії з білим ромбом на кінці;

– композиція: більш строгий варіант агрегації. При знищенні контейнера видаляється і його зміст. Відображається зв’язок у вигляді лінії з чорним ромбом на кінці;

– реалізація: це зв’язок між інтерфейсами та класами, що реалізують дані інтерфейси. Інтерфейс, як правило, реалізований у вигляді абстрактного класу. Відображається зв’язок у вигляді пунктирної лінії з трикутною стрілкою в кінці [9].

Діаграма класів для проєктування системи реалізована для прецеденту Керування робочим процесом та моніторинг. Для її створення було виділено основні сутності, які було відображено у вигляді класів, додано атрибути з вказанням їх типізації та операції, пов’язані з даними класами. Далі було визначено типи зв’язків між класами та додано навігацію. Створена діаграма класів зображена на рисунку 2.4.

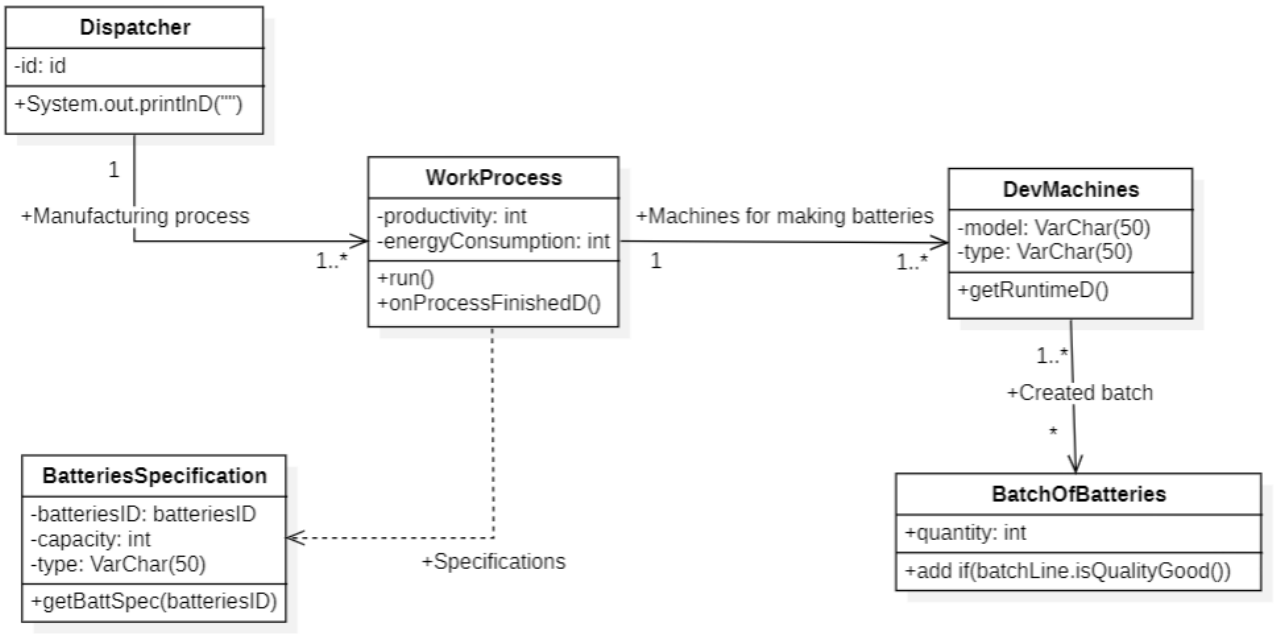

Рисунок 2.4 – Розроблена діаграма класів для проєктування компоненту системи

2.3 Опис основних процесів виробництва телефонних акумуляторів та графіку його роботи

При створенні якісної та надійної системи MES потрібно враховувати, якими є основні виробничі процеси конкретного підприємства та як правильно

підлаштовувати систему під них. Алгоритм виготовлення акумуляторів має такі етапи:

– замішування пасти (суспензії) – процес включає в себе підготовку активних матеріалів (літій-іонних з'єднань) у вигляді пасти або суспензії. Вона складається з активного матеріалу (наприклад, вуглець), в'яжучої речовини (наприклад, PVDF – полівініліденфторид) та розчинника (зазвичай це NMP – N-метилпіролідон). Ці компоненти змішуються в міксерах, щоби отримати однорідну масу, необхідну для подальшого нанесення на колектори;

– замішування електроліту – процес включає в себе з'єднання солей літія (наприклад, LiPF6), з органічними розчинниками. Замішування відбувається в в умовах високої стерильності, щоб знизити ризик попадання різного виду елементів, які можуть вплинути на працездатність акумулятора;

– нанесення пасти на колектори – паста наноситься на металічні колектори (алюмінієві фольги для катодів та мідні – для анодів) тонким та однорідним шаром товщиною декілька міліметрів;

– сушіння електродів – після нанесення пасти електронні смуги піддаються просушуванню для видалення розчинника. Процес відбувається у сушильних печах при контрольованій температурі, щоби забезпечити повне видалення розчинника та зберегти структуру активного матеріалу;

– каландрування електродів – це процесс прокатки електродних смуг через гідравлічний прес для зменшення їх товщини та підвищення густини. Це покращує контакт між частками активного матеріалу та підвищує провідність, що позитивно впливає на продуктивність акумулятора. Оптимальною товщиною електродів після стискання є 120 мкм, тому ступінь стискання повинна бути $\acute{\varepsilon} = 20$ % [9];

– нарізання електродів – після каландрування смуги електродів нарізаються на окремі частини, які будуть використовуватися при збиранні акумулятора. Нарізання проводиться з високою точністю, щоб забезпечити правильне співвідношення розмірів анодів та анодів.

– укладання елементів електродів – процес, при якому електроди вкладаються у послідовності анод-сепаратор-катод. Сепаратор потрібен для того, щоб унеможливити ризик короткого замикання при прямому контакті катода з анодом, але в той самий час дозволити іонам літія вільно переміщуватися між ними;

– заповнення електролітом – процес заповнення корпусу акумулятора електролітом відбувається в умовах вакууму задля забезпечення гарного просочування електролітом всіх пор електродів та сепаратора;

– запаювання батареї – процес, при якому корпус акумулятора герметизується задля запобігання витоку електроліту та проникнення повітря або вологи. Для цього найчастіше використовуються методи лазерного або ультразвукового зварювання.

Також було визначено, що підприємство буде працювати 8 годин на день, зі змінами з 8:00 до 12:00 та з 12:00 до 16:00. Таке рішення було прийнято з огляду на те, що робота компоненту MES-системи Управління виробничим процесом буде більш зрозумілою, якщо демонструвати її на прикладі невеликого підприємства.

2.4 Опис розрахунків даних для імітаційного моделювання системою процесів виробництва

Для правильного і точного моделювання системою процесів у вигляді графіків їй необхідно надати деякі числові дані, що вносяться до формул для розрахунків. Для деяких машин формули розрахунків різняться через особливості роботи даного обладнання. Для розрахунку та відображення на графіку кількості часу (у годинах), який витрачає машина на виготовлення всіх партій деталей, запланованих на поточний день, необхідно використовувати формулу:

$$R = (a \times N) + (a_1 \times N_1) \dots + (a_n \times N_n), \quad (2.1)$$

де R – час, який витрачає машина на виготовлення партій деталей, год;

a, a_1, a_n – час, який витрачає машина на виготовлення одного акумулятора, год;

N, N_1, N_n – кількість деталей у партії.

Якщо машини є міксерами для замішування суспензії або електроліту, то формула для них відрізняється:

$$R = (y \times (m \times N)) + (y_1 \times (m_1 \times N_1)) \dots + (y_n \times (m_n \times N_n)) , \quad (2.2)$$

де y, y_1, y_n – час, за який міксер вимішує один літр пасти або електроліту, год;

m, m_1, m_n – кількість пасти або електроліту, яка потрібна для одного акумулятора, кг;

Для точності розрахунків та правильного створення програм кожну змінну у формулах було знайдено для кожного процесу виготовлення окремо.

2.4.1 Розрахунки для міксера з вимішування активного матеріалу (суспензії)

Для підстановки даних у загальну формулу (згідно з формулою (2.2)) необхідно розрахувати значення кількості пасти, яка потрібна для одного акумулятора та час, за який міксер вимішує один літр пасти. Для цього необхідно розраховувати значення окремо для катода і анода. Для розрахунків знадобляться розміри катодів та анодів літій-іонних акумуляторів. Значення були обрані з огляду на стандартний діапазон (від 70 мм до 80 мм довжина, та від 50 мм до 60 мм – ширина), використовуваний для виготовлення акумуляторів даного типу та проведення досліджень над ними [10]. Також будуть необхідні значення густини активного матеріалу та товщини нанесення суспензії на електроди. З огляду на те, що в якості матеріалу для катода використовується LiCoO2, його густина становить 4,9 г/см³. В якості матеріалу для анода використовують графіт, тому його густина буде 2,26 г/см³ [11]. Товщина нанесеного шару суспензії зазвичай однакова для катода і анода та має статичне значення 100 мкм (0,01 см) [12]. Кроки розрахунку:

Спочатку буде розраховано масу пасти для катода. Для розрахунку площі катода (для двох сторін) візьмемо значення довжини 80 мм (8 см) та ширини 60 мм (6 см).

$$S = 2 \times (l \times b) = 2 \times (8 \times 6) = 96 \text{ см}^2,$$

де S – площа електрода, см^2;

l – довжина електрода, см;

b – ширина електрода, см.

Далі буде розраховано значення об'єму активного матеріалу, виходячи зі знайденого значення площі:

$$V = S \times z = 96 \times 0{,}01 = 0{,}96 \text{ см}^3,$$

де V – об’єм активного матеріалу, см^3;

z – товщина активного матеріалу, см.

Далі буде розраховано масу суспензії:

$$m = V \times \rho = 0{,}96 \times 4{,}9 = 4{,}704 \text{ г},$$

де m – маса суспензії, г;

ρ – густина матеріалу LiCoO2, г/см³.

Тепер буде розраховано масу пасти для анода. Площу та об’єм він має таку саму, як і катод, тому розрахуємо одразу масу:

$$m = V \times \rho = 0{,}96 \times 2{,}26 = 2{,}17 \text{ г}.$$

Отже, на виготовлення одного літій-іонного акумулятора необхідно 6,874 грамів пасти, це 0,006874 кілограма.

Тепер розрахуємо час, за який міксер може вимісити один літр пасти. Для цього необхідно буде використати дані безпосередньо з машини, а точніше швидкість вимішування (об/хв) та максимальну кількість пасти, яку вона може вміщувати за раз. Для розрахунків було взято виробничий спеціалізований міксер для вимішування суспензії TOB-XFZH05 [13]. Швидкість вимішування ним пасти сягає 60 оборотів/хвилину, а максимально він може вміщувати в себе 2 літри активного матеріалу. Розрахуємо об'єм пасти, яка перемішується за один оборот:

$$V = \frac{e}{s} = \frac{2}{60} = \frac{1}{30} \text{ л},$$

де V – об'єм пасти, яка вимішується за один оборот, л.

e – місткість міксера, л;

s – швидкість вимішування пасти, об/хв.

Отже, для вимішування одного літру суміші знадобиться 30 оборотів міксера. Тепер можемо оцінити, що якщо міксер робить на хвилину 60 оборотів, то для 30 оборотів йому знадобиться 0,5 хвилини, це 0,0083 години.

2.4.2 Розрахунки для міксера з вимішування електроліту

Для підстановки даних у загальну формулу (згідно з формулою (2.2)) необхідно розрахувати значення кількості пасти, яка потрібна для одного акумулятора та час. Час, за який міксер вимішує один літр пасти, не потрібно додатково розраховувати, тому що міксери для суспензії та електроліту мають однакові параметри швидкості обертання та максимального розміру замісу.

Для розрахунку маси необхідно знати густину електроліту, об'єм електроліту на 1 Вт-годину, напругу акумулятору та його ємність. Значення були обрані, опираючись на стандартні діапазоні значень для літій-іонних акумуляторів, які найчастіше використовуються у роботі з ними та дослідженнях [14]. Отже, густина електроліту складає 1,2 г/моль (без додаткових домішок та сумішей), об'єм

електроліту – 0,3 мл на 1 Вт/годину ємності акумулятора, напруга зазвичай у літій-іонних акумуляторів складає від 3,3 В до 3,7 В. В якості ємності акумулятору було взято стандартне значення для сучасних телефонних акумуляторів, а саме 5000 мАч. Спочатку необхідно перевести ємність акумулятора з мАч у Вт/годину:

$$C = \frac{c \times U}{1000} = \frac{5000 \times 3{,}7}{1000} = 18{,}5 \text{ Вт/год},$$

де C – ємність акумулятора, Вт/год;

c – ємність акумулятора, мАч;

U – напруга акумуляторів, В.

Потім розрахувати об’єм електроліту за визначеною раніше ємністю:

$$V = C \times V_1 = 18{,}5 \times 0{,}3 = 5{,}55 \text{ мл},$$

де V – об’єм електроліту на 18,5 Вт/годину, мл;

V_1 – об’єм електроліту на 1 Вт/годину, мл.

Потім визначити масу електроліту для одного акумулятора, використовуючи густину електроліту:

$$m = V \times \rho = 5{,}55 \times 1{,}2 = 6{,}66 \text{ г},$$

де m – маса електроліту для одного акумулятора, г;

ρ – густина електроліту, г/см^3.

Отже, для одного акумулятора потрібно 6,66 грамів електроліту, в кілограмах це 0,00666 кг.

2.4.3 Розрахунки для машини з нанесення пасти на електроди

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина обробляє один акумулятор. Для цього необхідно знати такі дані: швидкість нанесення пасти машиною, площу поверхні електродів та товщину шару нанесення.

Площа поверхні електродів була розрахована раніше, тому S = 96 см^2. Товщина шару нанесення, як зазначалося раніше, однакова для шару катода і анода і дорівнює 100 мкм (0,01 см). Для знаходження швидкості нанесення пасти була взята спеціалізована машина TOB-JS350-3.0 [15], тому значення швидкості дорівнює від 0 м/хв до 1 м/хв (від 0 см/хв до 100 см/хв). Для розрахунків також знадобиться значення об'єму нанесеної пасти, яке дорівнює 0,96 см^3. Треба брати до уваги те, що машина покриває площу, рухаючись по довжині з визначеною шириною смуги. Ширина смуги дорівнює ширині електрода (6 см), тому що ширина покриття в машині може сягати до 10 см. Тоді довжина смуги буде дорівнювати 8 см (а враховуючи те, що нанесення необхідно з двох сторін, то загальна довжина дорівнює 16 см). Тоді час для двостороннього нанесення на один електрод буде розраховуватися:

$$t = \frac{l}{s} = \frac{16}{100} = 0{,}16 \text{ хв},$$

де t – час нанесення пасти на один електрод, хв;

l – довжина смуги нанесення, см;

s – швидкість нанесення, см/хв.

Отже, час обробки машиною обох електродів одного акумулятору дорівнює 0,32 хвилини (0,0053 години).

2.4.4 Розрахунки для сушильної печі

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який піч висушує електроди одного акумулятора. Для цього необхідно знати такі дані: температура сушіння електродів, товщина шару пасти та вид розчинника в пасті. Температура сушіння електродів була обрана зі стандартного діапазону значень (від 100 Cº до 150 Cº) для літій-іонних акумуляторів [16]. Товщина нанесеного шару суспензії була визначена раніше та має значення 100 мкм (0,01 см). Було обрано стандартний розчинник LiPF6, адже він найчастіше використовується у акумуляторах даного типу [17]. Для огляду була взята спеціалізована піч для сушіння електродів DZF-6050 [18].

Акумулятор має доволі тонкий шар нанесеної пасти, тому, з огляду на характеристики печі, для випаровування розчинника з неї необхідно витримувати електроди в печі від 30 хв до 40 хв. Опираючись на характеристики машини, було додатково визначено, що для підготовки електродів до сушіння необхідно 6 хвилин, для нагріву печі – 17 хвилин (беручи до уваги те, що вона нагрівається 7 Cº/хв), для охолодження та вилучення електродів з неї – 10 хвилин.

Отже, загальний час, за який піч висушує електроди одного акумулятора, дорівнює 68 хвилин (1,13 години).

2.4.5 Розрахунки для машини, яка проводить каландрування

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина каландрує електроди одного акумулятора. Для цього необхідно знати такі дані: швидкість прокатки електродів машиною та площу поверхні електродів, яка піддається каландруванню. Площа поверхні електродів була знайдена раніше, це 96 $см^2$, з неї було розраховано ширину електродів (6 см) та їх довжину (8 см). Прокатка відбувається одразу з двох сторін (між двома валиками), тому довжину візьмемо 8 см. Для визначення швидкості прокатки було взято спеціалізовану машину для каландрування DR-H150-200 [19].

Її швидкість дорівнює 2 м/хв (200 см/хв). Тепер можна визначити час, необхідний для прокатки електроду:

$$t = \frac{l}{s} = \frac{8}{200} = 0{,}04 \text{ хв},$$

де t – час прокатки електроду, хв;

s – швидкість прокатки, см/хв.

Отже, загальний час, за який буде каландровано всі електроди акумулятора, дорівнює 0,08 хвилин (0,0013 годин).

2.4.6 Розрахунки для машини, яка проводить розділення електродів

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина розрізає електроди одного акумулятора. Для цього необхідно знати такі дані: швидкість розділення машиною електродів та довжину одного електрода. Довжина, як було зазначено раніше, дорівює 8 см. Для знаходження швидкості нарізання було взято спеціалізовану машину TOB-MSK-300 [20]. Її швидкість дорівнює 3 м/хв (300 см/хв). Для розрахунку часу розділення двох електродів було:

$$a = \frac{l}{s} = \frac{8}{300} = 0{,}026 \text{ хв},$$

де a – загальний час, витрачений на розділення всіх електродів по довжині, хв;

s – швидкість нарізання електродів, см/хв.

Отже, машина розділить електроди одного акумулятора за 0,026 хвилин, або 0,00043 години.

2.4.7 Розрахунки для машини, яка вкладає частини акумулятора у герметичний блок

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина укладає всі складові одного акумулятора (катод, сепаратор, анод). Структура блоку укладки включає в себе: струмоприймач (алюмінієвий) – активний матеріал (анод) – сепараторний лист – активний матеріал (катод) – струмоприймач (мідний) [21]. Для розрахунку часу знадобиться швидкість укладання машиною елементів. В якості машини візьмемо ADP-300B [22], її швидкість укладання дорівнює 1,4 с/шт. З огляду на це, загальний час обробки машиною одного акумулятора буде дорівнювати 7 секунд (0,002 години).

2.4.8 Розрахунки для вакуумної машини з заливки електроліту

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина заливає електроліт в один акумулятор. Для точного розрахунку необхідно враховувати деякі етапи та час їх проходження. Для знаходження параметрів було обрано вакуумну машину TOB-ZYJ-02 [23]. Час проходження етапів було взято зі стандартих діапазонів даних, які використовуються в дослідженнях процесу заливки електроліту [24].

Етапи: розміщення акумулятора у вакуумній камері (займає приблизно 1-2 хвилини); створення вакууму в камері (у машині він 0,1 МПа) займає 2 хвилини; процес введення електроліту займає 60 секунд (на один акумулятор, як було розраховано раніше, необхідно 6,66 грамів електроліту, а максимальний об’єм однієї заливки у машини – 7 грамів, тому впорскування потрібної кількості відбувається за один раз); вирівнювання тиску та вилучення акумулятору займає близько 90 секунд.

Тоді загальний час, затрачений на заливку електроліту в один акумулятор, буде дорівнювати 6 хвилин (0,1 години).

2.4.9 Розрахунки для вакуумної машини з герметизації блоку акумулятора

Для підстановки даних у загальну формулу (згідно з формулою (2.1)) необхідно розрахувати час, за який машина повністю герметизує один блок. Процес герметизації проводиться таким чином: розміщення акумулятора у камері; створення вакууму; процес герметизації (з використанням лазерного зварювання); вирівнювання тиску та вилучення батареї. Для розрахунків буде використана машина TOB-YF200-JZ [25]. Час розміщення акумулятора у вакуумі в основному займає від 1 хвилини до 2 хвилин, створення вакууму (машині потрібно 98 КПа) займе приблизно 60 секунд, безпосередньо зварювання, як вказано у технічній карті машини, займе 50 секунд. Час на вилучення акумулятору і повернення до нормального тиску (з урахуванням ступеню вакууму) займе близько 60 секунд.

Тоді загальний час, затрачений на заливку електроліту в один акумулятор, буде дорівнювати 4 хвилини (0,06 години).

2.5 Охорона праці

У сучасному виробничому середовищі шум став невід'ємною складовою багатьох технологічних процесів. Проте, його вплив на здоров'я та самопочуття працівників не можна недооцінювати. Високий рівень шуму не тільки заважає спокійному робочому процесу, але й може стати причиною серйозних проблем зі здоров'ям. Саме тому контроль і управління рівнем шуму на підприємстві є ключовими аспектами охорони праці. Вплив тривалого високого рівня шуму може призвести до таких наслідків, як:

– втрата слуху: тривале перебування в умовах високого рівня шуму може призвести до незворотних змін у слуховій системі працівників, таких як шумова індукована втрата слуху;

– стрес та втома: постійний шум може викликати стрес, підвищення рівня тривоги та фізичну втому, що знижує загальну продуктивність праці;

– порушення концентрації: шум заважає концентрації та точності виконання завдань, що може підвищити ризик виробничих травм та аварій.

Для забезпечення безпеки працівників встановлені певні нормативи рівня шуму на робочих місцях. Згідно з нормативами, загальний рівень шуму машин на виробництві з виготовлення акумуляторів повинен бути у наступних діапазонах: міксер для вимішування суспензії – від 78 дБ до 80 дБ; міксер для вимішування електроліту – від 75 дБ до 80 дБ; машина для нанесення – від 70 дБ до 72 дБ; піч – від 74 дБ до 80 дБ; гідравлічний прес – від 90 дБ до 94 дБ; лазерний різак – від 70 дБ до 74 дБ; машина для вкладання – від 70 дБ до 80 дБ; машина для заливки електроліту – від 70 дБ до 85 дБ; машина для герметизації – від 75 дБ до 85 дБ [26]. При перевищенні цього рівня необхідно вживати заходи для його зниження або забезпечити працівників засобами індивідуального захисту слуху. Методи зниження рівня шуму:

– технічні заходи: використання звукоізоляційних матеріалів, встановлення шумозахисних екранів, модернізація обладнання.

– організаційні заходи: раціоналізація виробничих процесів, зменшення тривалості перебування працівників у зонах підвищеного шуму.

– засоби індивідуального захисту: використання спеціальних навушників або вушних вкладишів для захисту слуху.

– вчасна перевірка коректності роботи машини: при раптових підвищеннях показника рівня шуму машини необхідно проводити діагностику та перевіряти працездатність її складових.

Розрахунок рівня шуму на підприємстві можна здійснити, використовуючи методи акустичних вимірювань. Однією з базових формул для оцінки сукупного рівня шуму є:

$$L_{eq} = 10 \log_{10} \left(\frac{1}{T} \sum_{i=1}^{n} T_i \times 10^{\frac{L_i}{10}} \right) \quad (2.3)$$

де L_{eq} – еквівалентний рівень шуму, дБ(А);

T – загальний час вимірювання, с;

T_i – час, протягом якого рівень шуму дорівнює L_i, с;

L_i – рівень шуму, виміряний на i-тому інтервалі, дБ(А);

n – кількість інтервалів вимірювання.

Забезпечення правильного рівня шуму на підприємстві є важливою складовою охорони праці. Дотримання нормативів та впровадження ефективних заходів для зниження шуму сприяє збереженню здоров'я працівників, підвищенню їх продуктивності та загальному покращенню умов праці.

3 СТВОРЕННЯ БАЗИ ДАНИХ ПРЕДМЕТНОЇ ОБЛАСТІ

3.1 Поняття ER-моделювання

Модель сутність-зв'язок (англ. Entity-relationship model або entity-relationship diagram) – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель – це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, що його реалізує, є модель сутність-зв'язок. Модель сутність-зв'язок є результатом систематичного процесу, який описує та визначає деяку предметну область. Вона не визначає сам процес, а лише візуалізує його. Дані представлені у вигляді компонентів (сутностей), які пов'язані між собою певними зв'язками, які виражають залежності і вимоги між ними. Сутності можуть мати різні властивості (атрибути), які характеризують їх. Діаграми, створені для представлення цих сутностей, атрибутів і зв'язків графічно, називають сутність-зв'язок діаграмами.

ER-модель зазвичай реалізується в вигляді баз даних. У разі реляційної бази даних, в якій зберігаються дані в таблицях, кожен рядок кожної таблиці являє собою один екземпляр, деякі поля даних в цих таблицях вказують на індекси в інших таблицях – вони є покажчиками фізичної реалізації зв'язків між сутностями [28].

Для великих баз даних побудова ER-моделі дозволяє уникнути помилок проєктування, які надзвичайно важко виправляти, особливо, якщо база даних вже експлуатується чи на стадії тестування. Помилки в розробці структури бази даних може призвести до перебудови коду програмного забезпечення, що керує цією базою даних [29].

3.2 Створення таблиць бази даних

Керуючись наведеним у пункті 2.3 алгоритмом виготовлення акумуляторів та UML-діаграмами, описаними у пункті 2.2, було створено ER-модель бази даних (рис. 3.1) у застосунку для взаємодії з базами даних PhPMyAdmin.

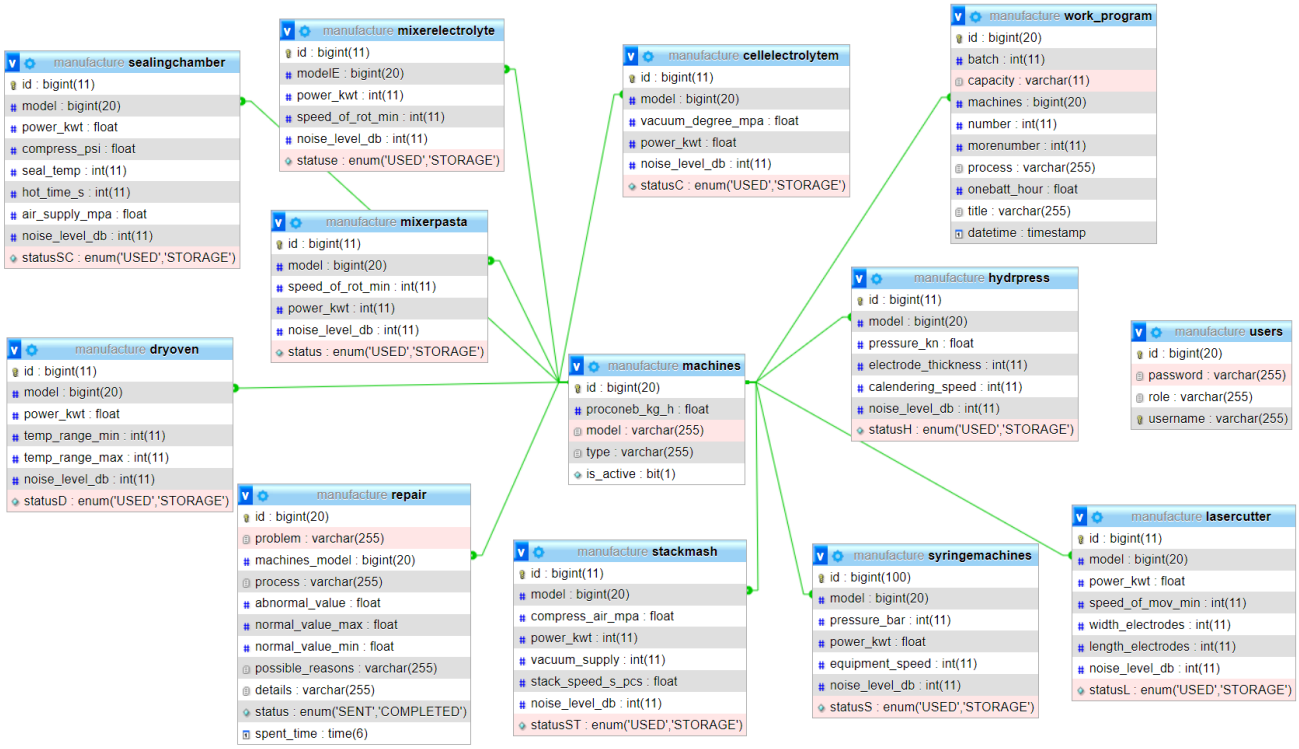

Рисунок 3.1 – Розроблена модель бази даних

3.2.1 Опис зв’язків

Опис зв’язків між сутностями системи наведено у таблиці 3.1.

Таблиця 3.1 – Опис зв’язків

Сутність 1	Зв’язок	Сутність 2	Тип зв’язку
machines	знаходиться у	mixerpasta	(1:1)
machines	знаходиться у	mixerelectrolyte	(1:1)
machines	знаходиться у	syringemachines	(1:1)
machines	знаходиться у	dryoven	(1:1)
machines	знаходиться у	hydrpress	(1:1)
machines	знаходиться у	lasercutter	(1:1)
machines	знаходиться у	stackmash	(1:1)
machines	знаходиться у	cellelectrolytem	(1:1)
machines	знаходиться у	sealingchamber	(1:1)
machines	міститься у	repair	(1:M)
machines	міститься у	work_program	(1:M)

Опис зв’язків:

– одна модель машини може бути лише у одного міксера для замішування пасти, один міксер може мати лише одну модель;

– одна модель машини може бути лише у одного міксера для замішування електроліту, один міксер може мати лише одну модель;

– одна модель машини може бути лише у однієї машини, одна машина може мати лише одну модель;

– одна модель машини може бути лише у однієї печі, одна піч може мати лише одну модель;

– одна модель машини може бути лише у одного гідравлічного пресу, один прес може мати лише одну модель;

– одна модель машини може бути лише у одного лазерного різака, один різак може мати лише одну модель;

– одна модель машини може бути лише у однієї машини для кладання електродів, одна машина для укладання електродів може мати лише одну модель;

– одна модель машини може бути лише у однієї машини для заливки електроліту, одна машина може мати лише одну модель;

– одна модель машини може бути лише у однієї машини для герметизації блоку акумулятора, одна машина може мати лише одну модель;

– одна модель машини може міститися у декількох рапортах, один рапорт складається лише на одну машину;

– один тип машини може бути у декількох робочих програмах, одна робоча програма може містити лише один тип машини.

3.2.2 Опис таблиці mixerpasta

Дана таблиця містить інформацію про міксер для замішування суспензії (пасти) та детальні параметри для його злагодженої роботи. Складається вона з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– speed_of_rot_min (INT) – числове значення, описує мінімальне значення швидкості, з якою обертаються лопаті міксера;

– power_kwt (INT) – числове значення, описує потужність, з якою працює міксер;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого міксера;

– status (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює вона чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.2.

id	model	speed_of_rot_min	power_kwt	noise_level_db	status
1	1	60	1	78	STORAGE
2	10	70	3	70	USED
3	11	40	3	79	STORAGE

Рисунок 3.2 – Вміст таблиці mixerpasta

3.2.3 Опис таблиці mixerelectrolyte

Дана таблиця містить інформацію про міксер для замішування електроліту та детальні параметри для його злагодженої роботи. Складається вона з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (INT) – числове значення, описує потужність, з якою працює міксер;

– speed_of_rot_min (INT) – числове значення, описує мінімальне значення швидкості, з якою обертаються лопаті міксера;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого міксера;

– statusE (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює вона чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.3.

id	modelE	power_kwt	speed_of_rot_min	noise_level_db	statuse
1	2	3	70	80	USED
2	12	67	3522	75	STORAGE

Рисунок 3.3 – Вміст таблиці mixerelectrolyte

3.2.4 Опис таблиці syringemachines

Дана таблиця містить інформацію про машини для нанесення суспензії на електроди та детальні параметри для їх злагодженої роботи. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– pressure_bar (INT) – числове значення, описує ступінь тиску на поверхню при нанесенні пасти, вимір у барах;

– power_kwt (FLOAT) – числове значення, описує потужність, з якою працює машина;

– equipment_speed (INT) – швидкість обладнання, з якою машина наносить пасту;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючої машини;

– statusS (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює вона чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.4.

id	model	pressure_bar	power_kwt	equipment_speed	noise_level_db	statusS
1	3	16	0.03	3	70	USED
3	20	17	0.05	3	72	STORAGE

Рисунок 3.4 – Вміст таблиці syringemachines

3.2.5 Опис таблиці dryoven

Дана таблиця містить інформацію про виробничі печі для просушування електродів після нанесення пасти та детальні параметри для їх злагодженої роботи. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (FLOAT) – числове значення, описує потужність, з якою працює піч;

– temp_range_min (INT) – мінімальне значення діапазону температури, в якому має працювати піч;

– temp_range_max (INT) – максимальне значення діапазону температури, в якому має працювати піч;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючої печі;

– statusD (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус печі, працює вона чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.5.

id	model	power_kwt	temp_range_min	temp_range_max	noise_level_db	statusD
1	8	1.4	10	250	74	USED
2	18	12	10	200	79	STORAGE

Рисунок 3.5 – Вміст таблиці dryoven

3.2.6 Опис таблиці hydrpress

Дана таблиця містить інформацію про гідравлічний прес для процесу каландрування електродів та детальні параметри для його злагодженої роботи. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– pressure_kn (FLOAT) – параметр для позначення тиску на поверхню електродів, вимірюється у кілоньютонах (кН);

– electrode_thinkness (INT) – стале числове значення товщини електрода, до якої його потрібно стискати. Вимірюється у мікрометрах;

– calendering_speed (INT) – швидкість прокатки пресу по електродах;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого пресу;

– statusH (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус пресу, працює він чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.5.

id	model	pressure_kn	electrode_thickness	calendering_speed	noise_level_db	statusH
1	4	1.6	120	2	90	USED
2	14	1.6	120	1	94	STORAGE

Рисунок 3.5 – Вміст таблиці hydrpress

3.2.7 Опис таблиці lasercutter

Дана таблиця містить інформацію про лазерний різак для розділення електродів та детальні параметри для його злагодженої роботи. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (FLOAT) – числове значення, описує потужність, з якою працює різак;

– speed_of_mov_min (INT) – числове значення, яке описує швидкість, з якою різак проводить операції, вимірюється в метрах за хвилину;

– width_electrodes (INT) – статичне значення необхідної ширини електродів, якої повинен триматися різак при виконанні операцій. Вимірюється у міліметрах;

– length_electrodes (INT) – статичне значення необхідної довжини електродів, якої повинен триматися різак при виконанні операцій. Вимірюється у міліметрах;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого різака;

– statusL (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус різака, працює він чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.6.

id	model	power_kwt	speed_of_mov_min	width_electrodes	length_electrodes	noise_level_db	statusL
1	5	1	3	60	75	73	USED
2	15	1	2	60	75	76	STORAGE

Рисунок 3.6 – Вміст таблиці lasercutter

3.2.8 Опис таблиці stackmash

Дана таблиця містить інформацію про вакуумну машину для укладання електродів в герметичний блок та детальні параметри для злагодженої роботи машини. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (INT) – числове значення, описує потужність, з якою працює машина;

– compress_air_mpa (FLOAT) – числове значення, яке описує величину стиснутого повітря. Вимірюється у мегапаскалях (МПа);

– vacuum_supply (INT) – значення, яке описує максимально можливу подачу вакууму машиною. Вимірюється у літрах на хвилину;

– stack_speed_s_pcs (FLOAT) – описує швидкість укладання електродів. Вимірюється у секундах на штуку;

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого різака;

– statusST (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює він чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.7.

id	model	compress_air_mpa	power_kwt	vacuum_supply	stack_speed_s_pcs	noise_level_db	statusST
1	9	0.6	3	200	1.4	70	USED
2	19	0.5	4	210	1.3	78	STORAGE

Рисунок 3.7 – Вміст таблиці stackmash

3.2.9 Опис таблиці cellelectrolytem

Дана таблиця містить інформацію про вакуумну машину для заливання електроліту в герметичний блок та детальні параметри для злагодженої роботи машини. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (FLOAT) – числове значення, описує потужність, з якою працює машина;

– vacuum_degree_mpa (FLOAT) – числове значення, яке описує ступінь вакууму. Вимірюється у мегапаскалях (МПа);

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючого різака;

– statusC (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює він чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.8.

id	model	vacuum_degree_mpa	power_kwt	noise_level_db	statusC
1	7	0.1	1.5	70	USED
2	17	0.1	0.4	75	STORAGE

Рисунок 3.8 – Вміст таблиці cellelectrolytem

3.2.10 Опис таблиці sealingchamber

Дана таблиця містить інформацію про вакуумну машину для зварювання блоку батареї та детальні параметри для злагодженої роботи машини. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– model (BigInteger) – описує модель машини, зовнішній ключ таблиці machines;

– power_kwt (FLOAT) – числове значення, описує потужність, з якою працює машина;

– compress_psi (FLOAT) – числове значення, яке описує тиск ущільнення для кришки вакуумної камери. Вимірюється у параметрі фунт-сила на квадратний дюйм (PSI);

– seal_temp (INT) – числове значення, що описує температуру для зварювання. Вимірюється у градусах Цельсія;

– hot_time_s (INT) – числове статичне значення, яке описує час нагріву, вимірюється у секундах;

– air_supply_mpa (FLOAT) – числове значення, яке описує величину стиснутого повітря. Вимірюється у мегапаскалях (МПа);

– noise_level_db (INT) – числове значення, описує оптимальний рівень шуму для працюючої машини;

– statusSC (ENUM) – може приймати значення USED або STORAGE. Описує робочий статус машини, працює вона чи знаходиться на складі;

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.9.

id	model	power_kwt	compress_psi	seal_temp	hot_time_s	air_supply_mpa	noise_level_db	statusSC
1	6	1.2	60	-250	80	0.5	78	USED
2	16	0.05	50	-250	95	0.7	80	STORAGE

Рисунок 3.9 – Вміст таблиці sealingchamber

3.2.11 Опис таблиці machines

Дана таблиця містить інформацію про всі моделі машин та їх тип (для чого вони потрібні). Таблиця була створена для розвантаження системи та полегшення роботи проєктувальника, який може вільно обирати типи машин у вигляді списку з однієї таблиці, та диспетчера, який так само зі списку обиратиме моделі машин. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– proconeb_kg_h (FLOAT) – числове значення, описує, за скільки часу машина опрацює один акумулятор у годинах, або скільки потрібно суспензії чи електроліту на одну батарею в кілограмах (для міксерів);

– model (VarChar(255)) – строкове значення, описує моделі машин, використовуваних у підприємстві;

– type (VarChar(255)) – строкове значення, описує типи машин, тобто процеси, які вони виконують;

– is_active (bit(1)) – булеве значення, зберігає інформацію про те, чи активна зараз машина (чи працює вона на підприємстві).

Детальна інформація щодо вмісту таблиці наведена на рисунку 3.10.

id	proconeb_kg_h	model	type	is_active
1	0.006874	TOB-XFZH01	mixerpasta	0
2	0.00666	ElectroMixSBM-500	mixerelectrolyte	1
3	0.0053	TOB-JS350-3.0	syringemachines	1
4	0.0013	DR-H150-200	hydrpress	1
5	0.00043	TOB-MSK-300	lasercutter	1
6	0.06	TOB-YF200-JZ	sealingchamber	1
7	0.1	TOB-ZYJ-02	cellelectrolytem	1
8	1.13	DZF-6050	dryoven	1
9	0.002	TOB-ADP-300B	stackmash	1
10	0.0756	SuperMix1890	mixerpasta	1
11	0.094	MonstaX1900	mixerpasta	0
12	0.02	EleSuper100	mixerelectrolyte	0
14	0.0018	TOB-HRP-300	hydrpress	0
15	0.000024	MTechFL3019	lasercutter	0
16	0.19	TOB-BFZ-200-G	sealingchamber	0
17	0.117	TOB-ZY300	cellelectrolytem	0
18	1.2	TOB-DZF-6510	dryoven	0
19	0.02	TOB-S-DP-150	stackmash	0
20	0.00207	AKN800NHH0385	syringemachines	0

Рисунок 3.10 – Вміст таблиці machines

3.2.12 Опис таблиці repair

Дана таблиця містить інформацію про рапорти на ремонт деталей. Деякі колонки заповнюються лише диспетчером, а деякі – лише бригадиром ремонтної бригади. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– problem (VarChar(255)) – строкове значення, заповюється диспетчером. Містить інформацію про основну проблему, через яку створюється даний рапорт;

– machines_model (BigInteger) – числове значення, є зовнішнім ключем таблиці machines;

– process (VarChar(255)) – строкове значення, заповнюється диспетчером. Містить інформацію про процесс, при виконанні якого сталась поломка;

– abnormal_value (FLOAT) – числове значення, заповнюється диспетчером. Містить інформацію про числовий параметр, який не є прийнятним для роботи даної машини;

– normal_value_max (FLOAT) – числове значення, заповнюється диспетчером. Містить інформацію про нижню границю діапазону, який є прийнятним для злагодженої роботи машини;

– normal_value_max (FLOAT) – числове значення, заповнюється диспетчером. Містить інформацію про верхню границю діапазону, який є прийнятним для злагодженої роботи машини;

– possible_reason (VarChar(255)) – строкове значення, заповнюється диспетчером. Містить інформацію про процесс, при виконанні якого сталась поломка;

– details (VarChar(255)) – строкове значення, заповнюється бригадиром після проведення ремонту. Містить спростування або підтвердження припущення диспетчера щодо причини поломки, а також додаткові деталі ремонту;

– spent_time (time(6)) – значення часу, заповнюється бригадиром після проведення ремонту. Містить інформацію про те, скільки часу було витрачено на ремонт;

– status (ENUM) – статус рапорту. При створенні його диспетчером і відправці бригадиру статус є SENT. Після заповнення бригадиром усіх незаповнених полів статус стає COMPLETED.

Детальна інформація щодо вмісту таблиці не наводиться, тому що рядки не статичні та додаються користвачами вручну. Натомість на рисунку 3.11 наведено структуру таблиці.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	id	bigint(20)			Ні	*Немає*		AUTO_INCREMENT
☐	2	problem	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	3	machines_model	bigint(20)			Ні	*Немає*		
☐	4	process	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	5	abnormal_value	float			Ні	*Немає*		
☐	6	normal_value_max	float			Ні	*Немає*		
☐	7	normal_value_min	float			Ні	*Немає*		
☐	8	possible_reasons	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	9	details	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	10	status	enum('SENT', 'COMPLETED')	utf8_general_ci		Так	*NULL*		
☐	11	spent_time	time(6)			Так	*NULL*		

Рисунок 3.11 – Структура таблиці repair

3.2.13 Опис таблиці work_program

Дана таблиця містить інформацію про робочі програми, які створює проєктувальник на виробництво. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– batch (INT) – числове значення, яке містить інформацію про розмір партії акумуляторів для даної робочої програми;

– capacity (VarChar(11)) – містить у собі ємність батареї акумуляторів даної партії;

– machines (BigInteger) – означає використовуваний в роботі тип машини. Є зовнішнім ключем таблиці machines;

– number (INT) – числове значення, містить у собі номер замовлення, який було присвоєно клієнтові при отриманні від нього замовлення ішими працівниками підприємства;

– morenumber (INT) – числове значення, містить у собі номер партії одного замовлення, якщо вона була розбита на декілька частин проєктувальником;

– process (VarChar(255)) – строкове значення, означає процес, для якого створюється робоча програма;

– onebatt_hour (FLOAT) – числове значення, описує, за скільки часу машина опрацює один акумулятор у годинах, або скільки потрібно суспензії чи електроліту на одну батарею в кілограмах (для міксерів);

– title (VarChar(255)) – строкове значення, описує назву робочої програми;

– datetime (timestamp) – значення, яке містить у собі і дату, і час початку виконання даної робочої програми на виробництві;

Як і у випадку з таблицею repair, детальна інформація щодо вмісту таблиці не наводиться, тому що рядки не статичні та додаються користвачами вручну. Натомість на рисунку 3.12 наведено структуру таблиці.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	id	bigint(20)			Ні	*Немає*		AUTO_INCREMENT
☐	2	batch	int(11)			Ні	*Немає*		
☐	3	capacity	varchar(11)	utf8_general_ci		Так	*NULL*		
☐	4	machines	bigint(20)			Ні	*Немає*		
☐	5	number	int(11)			Ні	*Немає*		
☐	6	morenumber	int(11)			Так	*NULL*		
☐	7	process	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	8	onebatt_hour	float			Ні	*Немає*		
☐	9	title	varchar(255)	utf8_general_ci		Так	*NULL*		
☐	10	datetime	timestamp			Так	*NULL*		

Рисунок 3.12 – Структура таблиці work_program

3.2.14 Опис таблиці users

Дана таблиця містить інформацію про користувачів системою, їх логіни, паролі та роль. Таблиця складається з таких атрибутів:

– id (BigInteger) – унікальний ідентифікатор таблиці, первинний ключ;

– password (VarChar(20)) – пароль, який користувач вводить при авторизації;

– username (VarChar(100)) – логін, який користувач вводить при авторизації;

– role (VarChar(20)) – роль, яка перевіряється при авторизації користувача. Від неї залежить, на який інтерфейс програма переправить користувача.

Детальна інформація щодо вмісту таблиці наведена на рис. 3.13.

id	password	role	username
1	123456	ROLE_PLANNER	sofia@vt
2	234567	ROLE_DISPATCHER	elisa@vt
3	345678	ROLE_REPAIRER	oleh@vt

Рисунок 3.13 – Вміст таблиці users

3.3 Нормалізація таблиць баз даних

Процедура нормалізації БД полягає в усуванні надмірності даних та виявленні функціональних залежностей. Усунення надмірності даних гарантує компактність наборів даних за рахунок уникнення їх зайвого дублювання та унеможливлює виникнення аномалій вставки, вилучення й оновлення кортежів після фізичної реалізації БД.

Існують такі рівні нормалізації: перша нормальна форма (1НФ), 2НФ, 3НФ, нормальна форма Бойса-Кодда (БКНФ), 4НФ, 5НФ. Але на практиці використовують лише перші три рівня нормалізації – 1НФ, 2НФ, 3НФ.

Відношення буде зведено до першої нормальної форми (1НФ), коли всі його атрибути міститимуть тільки неподільні (атомарні) значення й у ньому відсутні групи атрибутів з однаковими за змістом значеннями, які повторюються у межах одного кортежу.

Відношення буде зведено до другої нормальної форми (2НФ), коли воно буде в першій нормальній формі, і кожний неключовий атрибут повністю визначиться первинним ключем, тобто щоб первинний ключ однозначно визначав кортеж і не був надлишковим (збігався із суперключем).

Відношення буде зведено до третьої нормальної форми (3НФ), коли воно буде у другій нормальній формі і у ньому не буде транзитивних залежностей між неключовими атрибутами, тобто значення будь-якого атрибута відношення, що не входить до первинного ключа, не залежить від значення іншого атрибута, що не входить до первинного ключа [30].

Для виконання завдання нормалізації необхідно привести базу даних до третьої нормальної форми. Для цього використано діаграми залежностей, які було складено для таблиць. Створені діаграми відображено на рисунках 3.14 – 3.26.

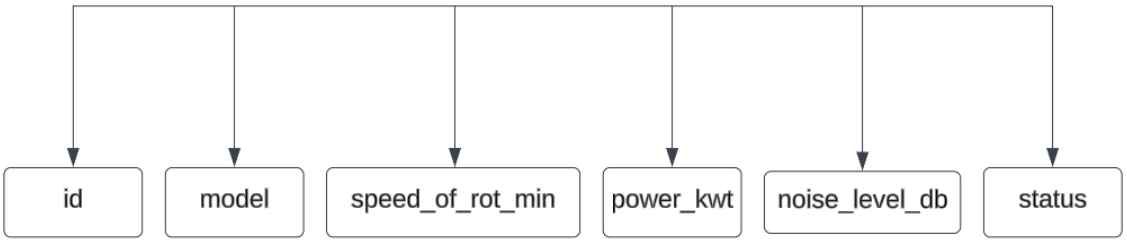

Рисунок 3.14 – Діаграма залежностей для таблиці mixerpasta

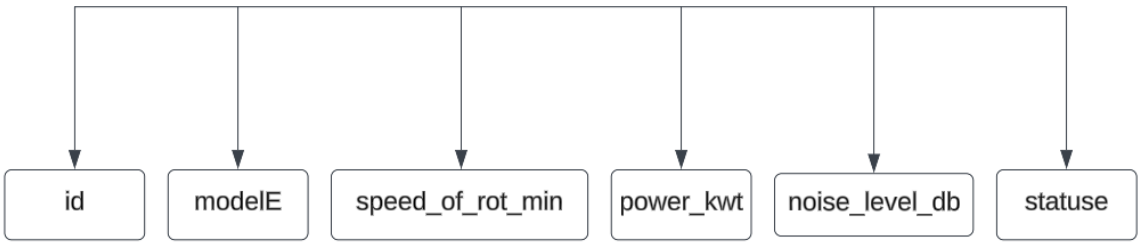

Рисунок 3.15 – Діаграма залежностей для таблиці mixerelectrolyte

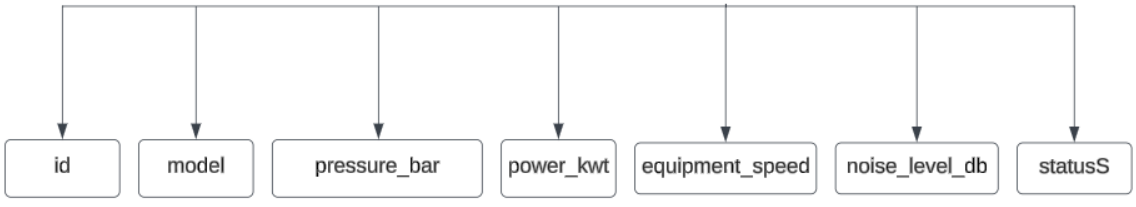

Рисунок 3.16 – Діаграма залежностей для таблиці syringemachines

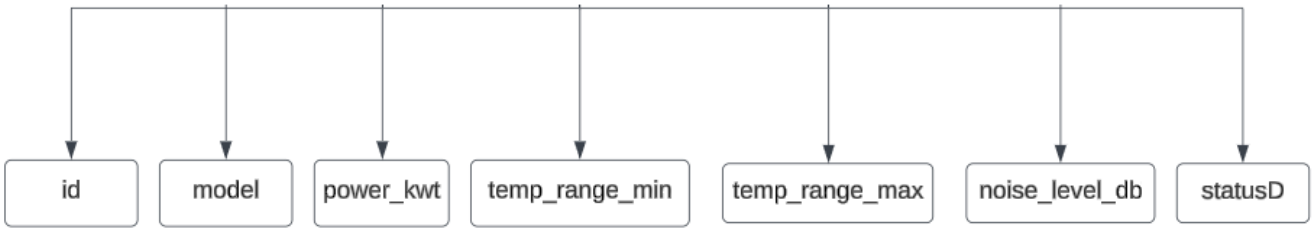

Рисунок 3.17 – Діаграма залежностей для таблиці dryoven

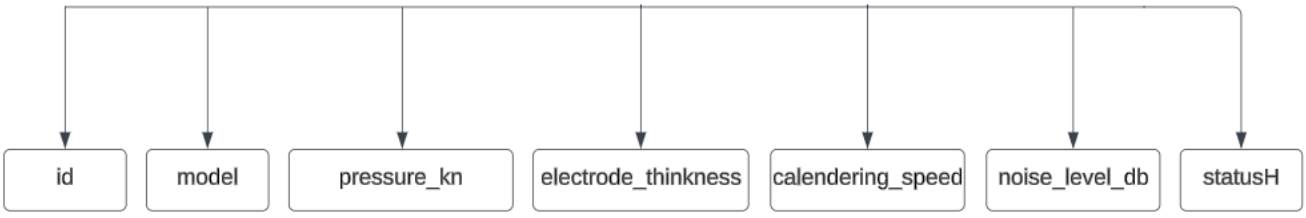

Рисунок 3.18 – Діаграма залежностей для таблиці hydrpress

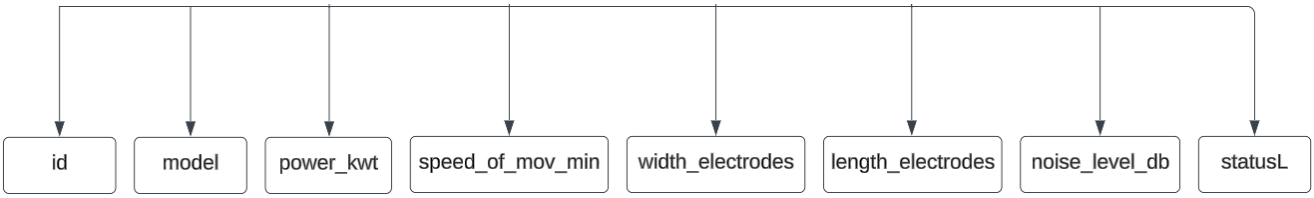

Рисунок 3.19 – Діаграма залежностей для таблиці lasercutter

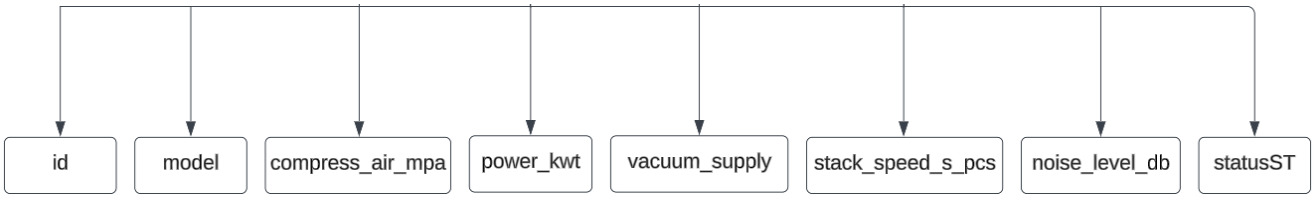

Рисунок 3.20 – Діаграма залежностей для таблиці stackmash

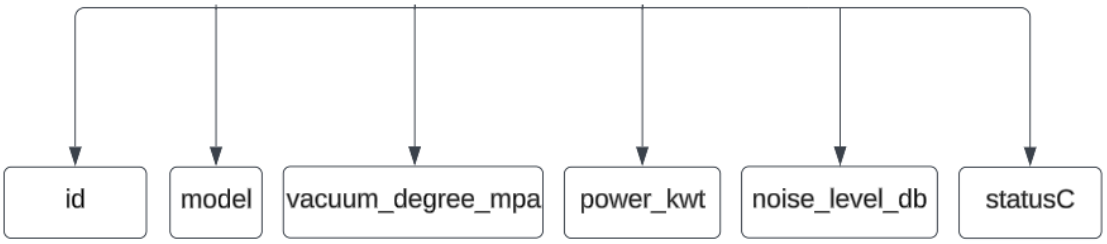

Рисунок 3.21 – Діаграма залежностей для таблиці cellelectrolytem

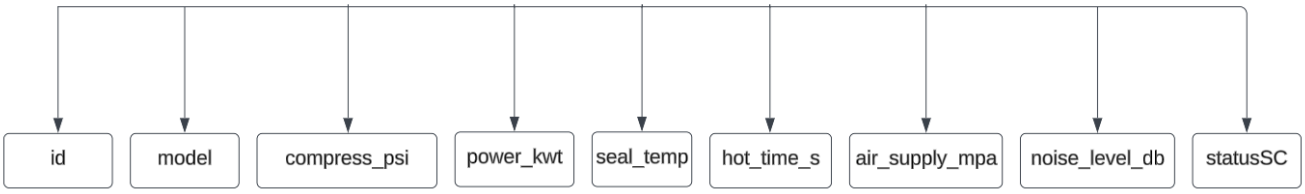

Рисунок 3.22 – Діаграма залежностей для таблиці sealingchamber

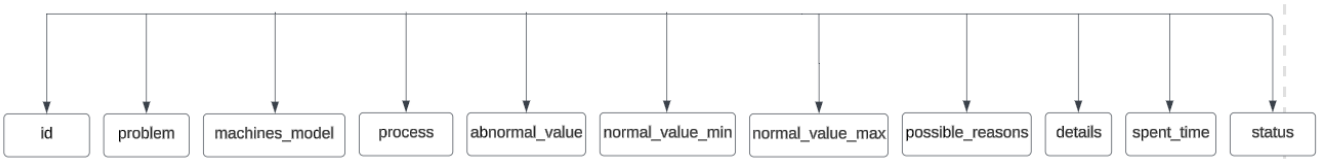

Рисунок 3.23 – Діаграма залежностей для таблиці repair

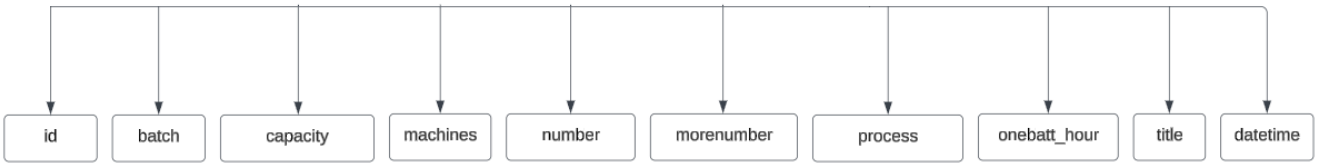

Рисунок 3.24 – Діаграма залежностей для таблиці work_program

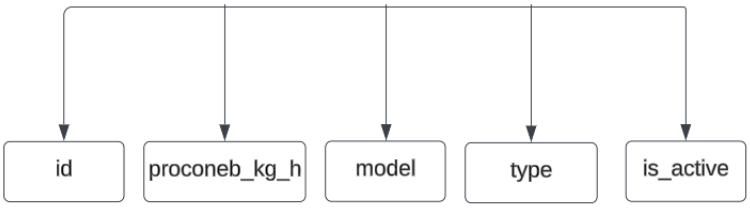

Рисунок 3.25 – Діаграма залежностей для таблиці machines

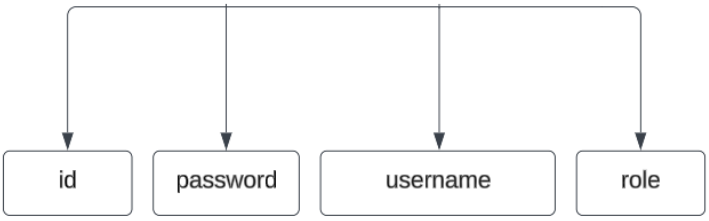

Рисунок 3.26 – Діаграма залежностей для таблиці users

Після проведення нормалізації можна зазначити, що дані таблиці приведені до першої нормальної формі, тому що в них визначені ключові атрибути, а інші атрибути залежать від первинного ключа своєї таблиці. Дані таблиці приведені до другої нормальної форми, бо вони приведені до першої нормальної форми і в них відсутні часткові залежності. Дані таблиці приведені до третьої нормальної формі, бо в них немає транзитивних залежностей.

4 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

4.1 Вибір мови програмування та допоміжного фреймворку

Для створення компоненту MES-системи для виробництва було використано мову Java. Java – об’єктно-орієнтована та крос-платформна мова програмування, випущена компанією Sun Microsystems у 1995 році (з 2009 року правами володіє компанія Oracle) [31]. Головними перевагами використання мови Java для програмування додатків є:

– об’єктно-орієнтоване програмування (ООП): Java заснований на принципах ООП, що сприяє високому рівні організації коду (структурування його у вигляді об’єктів), підвищеному рівню надійності (за рахунок інкапсуляції, яка приховує внутрішній код об’єктів, тим самим захищаючи їх від небажаної зміни), тощо;

– велика кількість бібліотек: Java пропонує великий набір стандартних бібліотек, які охоплюють широкий спектр функціональності – від роботи з колекціями до мережевої взаємодії та графічного інтерфейсу;

– багатопоточність: Java підтримує багатопоточність на рівні мови, що дозволяє створювати високопродуктивні багатозадачні додатки;

– активний розвиток та оновлення: мова регулярно оновлюється та розвивається, отримуючи новий функціонал, покращення продуктивності та безпеки. Засновник Oracle та спільнота активно підтримують та розвивають мову у всіх напрямках.

Також при створенні компоненту було задіяно фреймворк мови Java – Spring Boot. Для управління залежностями і збірки проєкту обрано Maven. Java Spring Boot - це засіб з відкритим вихідним кодом, який спрощує використання платформ на основі Java для створення мікрослужб та вебдодатків. Spring Boot виділяється серед інших фреймворків, оскільки він надає розробникам програмного забезпечення

гнучке налаштування, надійну пакетну обробку, ефективний робочий процес та велику кількість інструментів, допомагаючи розробляти надійні та масштабовані програми на базі Spring [32]. Головними перевагами використання Spring Boot є:

– автоконфігурація: фреймворк робить автоматичне налаштування конфігураційних файлів великої кількості бібліотек та фреймворків;

– спрощене керування залежностями: за допомогою Spring Boot можна швидко додати необхідні залежності до проєкту. Наприклад, spring-boot-starter-web – залежність, яка містить функціонал для створення вебдодатку, а spring-boot-starter-data-jpa надає можливість проєкту працювати з базами даних з використанням JPA (специфікація, яка визначає те, як об’єкти Java можуть бути пов’язані з реляційними базами даних);

– підтримка мікросервісів: в співпраці з Spring Cloud фреймворк може легко керуватия конфігурацією, виявленням сервісів, балансуванням навантаження та іншими аспектами мікросервісної архітектури;

– велика кількість допоміжної документації: Spring Boot має значну кількість пояснювальної документації, доступної через офіційні ресурси та спільноти розробників. Це значно полегшує вирішення складних питань та проблем при розробці.

Нижче наведено лістинг коду файлу pom.xml (рис. 4.1) з реалізацією підключення фреймворка та додаткових залежностей.

```
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.2</version>
    <relativePath/>     <!--     lookup     parent     from     repository     -->
</parent>
<groupId>com.example</groupId>
```

Рисунок 4.1 – Підключення Spring Boot та додаткових залежностей

```
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.2</version>
    <relativePath/>     <!--     lookup     parent     from     repository     -->
</parent>
<groupId>com.example</groupId>
<artifactId>Manufac</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Manufac</name>
<description>Manufacturing of something</description>
<properties>
    <java.version>17</java.version>
</properties>

<dependencies>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
       <groupId>mysql</groupId>
       <artifactId>mysql-connector-java</artifactId>
       <version>8.0.33</version>
       <scope>runtime</scope>
    </dependency>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-devtools</artifactId>
       <scope>runtime</scope>
       <optional>true</optional>
    </dependency>
    <dependency>
       <groupId>com.fasterxml.jackson.core</groupId>
       <artifactId>jackson-databind</artifactId>
    </dependency>
```

Рисунок 4.1, аркуш 2

```
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-test</artifactId>
       <scope>test</scope>
    </dependency>
</dependencies>
```

Рисунок 4.1, аркуш 3

4.2 Вибір СУБД та фреймворку для взаємодії з нею

При розробці системи була використана СУБД MySQL. MySQL – це система керування реляційною базою даних із відкритим кодом, яка широко використовується у розробці вебдодатків. MySQL надає багато функцій, таких як транзакції, індексація та безпека, що робить його хорошим вибором для вебдодатків, які потребують надійного керування даними. MySQL також забезпечує масштабовану архітектуру, яка може обробляти великі обсяги даних і запитів [33]. Основними перевагами використання MySQL є:

– висока продуктивність: СУБД здатна обробляти значні об’єми даних та велику кількість запитів одночасно за рахунок високого рівня її оптимізації. Також у ній наявні різні механізми зберігання інформації (таблиці InnoDB, MyISAM, тощо);

– масштабованість: MySQL може ефективно працювати як на невеликих сайтах з низьким рівнем навантаження, так і на крупних проєктах з мільйонами користувачів;

– надійність: у СУБД наявний механізм транзакцій InnoDB, який забезпечує підтримку ACID (атомарність, узгодженість, ізольованість, довговічність), що гарантує надійність операцій з даними. Також регулярні оновлення та активна підтримка спільноти розробників забезпечують виправлення помилок попереднії версій;

– безпека: наявна підримка багаторівневої безпеки, яка включає в себе права користувача, SSL-з’єднання, шифрування даних. Також є можливість самостійного налаштування подібних прав доступу для користувачів.

Для спрощення зв’язку класів Java з таблицями баз даних а також побудови запитів та обробки отриманих даних було використано ORM-фреймворк Hibernate.

ORM – це технологія програмування, яка пов’язує бази даних з концепціями об’єктно-орієнтованих мов програмування, створюючи віртуальну об’єктну базу даних. Це дозволяє розробнику працювати з базою даних, використовуючи поняття та принципи об’єктно-орієнтованого програмування замість створення SQL-запитів.

Hibernate – це фреймворк для об'єктно-реляційного відображення (ORM) у мові програмування Java, він забезпечує зручні засоби взаємодії з базами даних, представляючи дані у вигляді об'єктів, що дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід, нижче наведено основні коцепції Hibernate:

– сутності (Entities): сутності є об'єктами, які відображають таблиці в базі даних. Кожна сутність зазвичай відповідає окремій таблиці у базі даних;

– відносини між сутностями: Hibernate підтримує різні типи відносин між сутностями, такі як один до одного, один до багатьох і багато до багатьох;

– інструкції: анотації використовуються для мапінгу Java-класів на таблиці бази даних. Наприклад, анотація @Entity показує, що клас є сутністю;

– сесії (Sessions): сесії Hibernate є механізмом взаємодії з базою даних. Сесія управляє життєвим циклом сутності та виконує операції бази даних;

– HQL (Hibernate Query Language): HQL є об'єктно-орієнтованою мовою запитів, схожою на SQL, але використовує сутності Java замість таблиць бази даних;

– каскадування: каскадування дозволяє автоматично розповсюджувати операції (наприклад, збереження, оновлення, видалення) сутностей на пов'язані з ними сутності [34].

4.3 Вибір середовища програмної реалізації та додатків для взаємодії з базою даних та локальним сервером

В якості середовища розробки було обрано IntelliJ IDEA Ultimate Edition 2023.3.4. Воно є одним з найпопулярніших інтегрованих середовищ розробки для Java-проєктів та має низку переваг, що робить його найбільш вдалим вибором для створення додатків. Ключові можливості середовища розробки:

– розумне кодування: IntelliJ IDEA надає контекстно-залежні пропозиції по автодоповненню коду, що прискорює процес його написання та зменшує кількість помилок;

– підтримка різних фреймворків та систем збірки: IntelliJ IDEA підтримує основні Java-фреймворки, такі як Spring Boot, Hibernate, Java EE, пропонуючи спеціалізовані інструменти та конфігурації. Також воно легко інтегрується з Maven, Gradle та іншими системами збірки для спрощення керування залежностями та процесами збірки;

– наявність плагінів та налаштовуваних шаблонів: IntelliJ IDEA має широкий вибір плагінів, які розширюють функціональність IDE та дозволяють адаптувати її під конкретні потреби проєкта. Також є можливість створювати шаблони коду для прискорення написання часто використовуваних конструкцій.

Для роботи з базою даних було обрано вебзастосунок з відкритим кодом PhpMyAdmin. Він надає можливість адмініструвати сервер, запускати команди SQL та переглядати вміст таблиць через браузер. Для створення і запуску локального сервера на комп’ютері було використано програму MAMP. Для підключення

проєкту до локального сервера у файлі application.properties було написано такий код у рисунку 4.2:

```
spring.datasource.url=jdbc:mysql://localhost:3306/manufacture
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Рисунок 4.2 – Підключення проєкту до локального сервера

4.4 Налаштування системи авторизації користувачів

Для створення можливості авторизації користувачів було використано Spring Security. Це гнучкий фреймворк, розроблений для реалізації у вебдодатках аутентифікації, авторизації, захисту від різного типу атак за допомогою деяких токенів: CSFR (захист від несанкціонованого доступу), JWT (кодують інформацію про користувача та його права доступу), OAuth2 (використовуються для аутентифікації та авторизації через зовнішні провайдери, такі як Google, Facebook). Для взаємодії проєкту з фреймворком було підключено додаткові залежності у файл pom.xml (рис. 4.3).

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

Рисунок 4.3 – Підключення залежностей для роботи Spring Security

Потім було створено таблицю у базі даних users (її склад описано у пункті 3.2.14), куди занесено усі необхідні дані користувачів. В коді було створено репозиторій для взаємодії з даною таблицею та файл моделі User. Код репозиторію відображено на рисунку 4.4.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Рисунок 4.4 – Код репозиторію UsersRepository

В коді репозиторію реалізована функція findByUsername. Вона використовується у файлі UserService, який являє собою сервіс для керування користувачами у додатку. Дана функція необхідна для завантаження інформації про користувача по його логіну. При авторизації відбувається пошук введеного логіну у базі даних. Якщо його знайдено, то введений пароль кодується та створюється об’єкт CustomUserDetails, який містить інформацію про аутентифікованого користувача. Код UserService відображено на рисунку 4.5.

```
@Service
    public class UserService implements UserDetailsService {
        private final UserRepository userRepository;

        @Autowired
        private ApplicationContext applicationContext;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
```

Рисунок 4.5 – Код файлу UserService

```
        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        PasswordEncoder passwordEncoder =
applicationContext.getBean(PasswordEncoder.class);
        String encodedPassword =
passwordEncoder.encode(user.getPassword());


        User userWithEncodedPassword = new User(
                user.getId(),
                user.getUsername(),
                encodedPassword,
                user.getRole()
        );

        return new CustomUserDetails(userWithEncodedPassword);
    }
}
```

Рисунок 4.5, аркуш 2

CustomUserDetails реалізує інтерфейс UserDetails (інтерфейс Spring Security) та містить методи для отримання різноманітної інформації про користувача, необхідної для процесу аутентифікації. Він використовує методи, описані у моделі User (такі як getUsername(), getPassword()) для вилучення необхідних даних у необхідному форматі. Наприклад, для отримання ролі використовується вбудований у інтерфейс UserDetails метод getAuthorities, який повертає колекцію об’єктів, які являють собою ролі користувачів. Потім створюється об’єкт SimpleGrantedAuthority з потрібною роллю і переміщується у колекцію Collections.singleton. Код файлу CustomUserDetails представлено на рисунку 4.6.

```
public class CustomUserDetails implements UserDetails {
    private final User user;
    public CustomUserDetails(User user) {
        this.user = user;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
            return Collections.singleton(new
SimpleGrantedAuthority(user.getRole()));
    }
    @Override
    public String getPassword() {
        return user.getPassword();
    }
    @Override
    public String getUsername() {
        return user.getUsername();
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

Рисунок 4.6 – Код файлу CustomUserDetails

Після успішної авторизації використовується клас CustomAuthenticationSuccessHandler, який реалізовує інтерфейс AuthenticationSuccessHandler та використовується для обробки позитивного результату та перенаправлення користувача на необхідну сторінку в залежності від його ролі. Він отримує створену раніше колекцію GrantedAuthority, яка надає роль аутентифікованого користувача. Роль вилучається та оголошується змінна redirectUrl, яка буде містити URL-адресу, на яку буде направлено працівника в разі успішної авторизації. Код файлу CustomAuthenticationSuccessHandler наведено на рисунку 4.7.

```
public class CustomAuthenticationSuccessHandler implements
AuthenticationSuccessHandler {
    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response, Authentication authentication) throws
IOException, ServletException {
        Collection<? extends GrantedAuthority> authorities =
authentication.getAuthorities();
        String role = authorities.stream()
                .map(GrantedAuthority::getAuthority)
                .filter(authority -> authority.startsWith("ROLE_"))
                .findFirst()
                .orElse("");
        String redirectUrl;
        if (role.equals("ROLE_PLANNER")) {
            redirectUrl = "/orders";
        } else if (role.equals("ROLE_DISPATCHER")) {
            redirectUrl = "/disp-processes";
        } else if (role.equals("ROLE_REPAIRER")) {
            redirectUrl = "/subm-reports";
        } else {
            redirectUrl = "/default";
        }
        response.sendRedirect(redirectUrl); } }
```

Рисунок 4.7 – Код файлу CustomAuthenticationSuccessHandler

Також важливою складовою нормальної роботи авторизації необхідно реалізувати файл конфігурації SecurityConfig. Він містить правила авторизації для різних URL-адрес проєкту, налаштування форми входу до системи, виходу з неї, обробник успішної аутентифікації. Також SecurityConfig створює бін PasswordEncoder для роботи з паролями (їх кодування). Код файлу відображено на рисунку 4.8.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    private final UserDetailsService userDetailsService;
    public SecurityConfig(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception {
```

Рисунок 4.8 – Код файлу SecurityConfig

```
        http
                .authorizeHttpRequests((requests) -> requests
                        .requestMatchers("/", "/login", "/daily-
report").permitAll()
                        .requestMatchers("/orders","/program",
"/program/{id}/edit", "/program/{id}/remove",
                                "/reports", "/reports/{id}/pt", "/base-
plan").hasRole("PLANNER")
                        .requestMatchers("/disp-processes", "/disp-
processes/{id}/pt", "/monitoring", "/monitoring/repair", "/repair",
                                "/repair-info-disp").hasRole("DISPATCHER")
                        .requestMatchers("/subm-reports",
"/repair/{id}/update", "/replace-mach",
                                "/update-status", "/update-statusE",
"/update-statusS", "/update-statusD",
                                "/update-statusH", "/update-statusL",
"/update-statusST", "/update-statusC", "/update-
statusSC").hasRole("REPAIRER")
                        .requestMatchers("/img/**").permitAll()
                        .anyRequest().authenticated()
                )
                .formLogin((form) -> form
                        .loginPage("/login")
                        .failureUrl("/login?error=true")
                        .successHandler(new
CustomAuthenticationSuccessHandler())
                        .permitAll()
                )
                .logout((logout) -> logout
                        .logoutUrl("/logout")
                        .logoutSuccessUrl("/login")
                        .permitAll()
                );
        http.userDetailsService(userDetailsService);
        return http.build();
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Рисунок 4.8, аркуш 2

4.5 Розробка серверної частини системи

Для злагодженої роботи серверної частини системи було задіяно такі компоненти, як контролери (обробляють запити клієнта, виконують логіку додатку та повертають відповіді клієнту), моделі (структури даних та методи для взаємодії з БД, відповідають за складом таблицям з бази даних) та репозиторії (відповідають за доступ до даних БД, використовуються для виконання CRUD-операцій). Також було використано утиліти для виконання допоміжних задач в серверній

частині. Для зміни деяких значень у базі даних було реалізовано тригери, які діють при виконанні операцій на сервері.

4.5.1 Контролер ManufacController

Даний контролер створений для завантаження головної сторінки користувача Проєктувальник (рис. 4.9). За допомогою метода @GetMapping (анотація, яка використовується для вказівки на виконання запитів у визначеному URL) встановлюється шлях для виконання методу. Потім оголошується метод, який повертає рядок та приймає аргумент Model (model – аргумент для передачі даних з контролера в представлення). Далі прописано виклик, який додає атрибут в модель. Це означає, що на сторінці буде відображено атрибут з ім'ям title та значенням Main. Далі прописано значення метода, що повертається – це ім’я представлення, яке буде використано для рендерингу відповіді, тобто аргумент буде відображатися на шаблоні orders у даному випадку.

```
@Controller
public class ManufacController {
    @GetMapping("/")
    public String orders(Model model) {
        model.addAttribute("title", "Main");
        return "orders";
    } }
```

Рисунок 4.9 – Код контролеру ManufacController

4.5.2 Код контролеру ProgramController

Даний контролер створений для завантаження сторінки створення робочих програм проєктувальником. В ньому за допомогою анотації @Autowired було впроваджено репозиторії WorkProgramRepository та MachinesRepository для взаємодії з моделями WorkProgram та Machines і, відповідно, з таблицями бази даних work_program та machines. Далі за допомогою @GetMapping було

завантажено на сторінку список активних машин, згідно з методом, прописаним у репозиторії MachinesRepository (рис. 4.10).

Потім за допомогою анотації @PostMapping, яка використовується для зіставлення POST-запитів з методами контролера, отримано введені на сторінці сайту параметри, потім ці параметри за допомогою анотації @RequestParam вилучено та створено новий об’єкт WorkProgram, який збережено в базі даних. Після створення нової програми робиться перенос користувача на сторінку базового плану, де відображаються усі створені програми.

Далі прописано функціонал для сторінки редагування робочих програм. Тут за допомогою анотації @PathVariable вилучається параметр id визначеної програми, та за допомогою цього у базі даних знаходиться потрібна програма для редагування. При редагуванні програми вказано set-методи з моделей, це означає, що при завантаженні сторінки редагування потрібної програми в полях для вводу будуть вставлені старі значення. Після редагування зміни зберігаються у базі даних та користувача переадресовує на сторінку базового плану. Нижче реалізовано функціонал для видалення програми за допомогою операції delete.

```
@PostMapping("/program")
public String workProgramPost(@RequestParam int number, @RequestParam int
morenumber, @RequestParam String title, @RequestParam int batch, @RequestParam
String process, @RequestParam BigInteger machines, @RequestParam String capacity,
@RequestParam float onebatt_hour, @RequestParam String datetime, Model model) {
    Timestamp timestamp = Timestamp.valueOf(datetime);
```

Рисунок 4.10 – Код контролеру ProgramController

```
    WorkProgram WorkProgram = new
WorkProgram(number,morenumber,title,batch,process,machines,capacity,onebatt_hour,t
imestamp);
    WorkProgramRepository.save(WorkProgram);
    return "redirect:/base-plan";
}
@GetMapping("/program/{id}/edit")
public String editProgram(@PathVariable("id") Long id, Model model) {
    Optional<WorkProgram> optionalWorkProgram =
WorkProgramRepository.findById(id);
    if (optionalWorkProgram.isPresent()) {
        WorkProgram workProgram = optionalWorkProgram.get();
        List<Machines> activeModels = MachinesRepository.findActiveModels();
        model.addAttribute("workProgram", workProgram);
        model.addAttribute("machines", activeModels);
        return "program-edit";     } else {

        return "redirect:/base-plan";     }
}
@PostMapping("/program/{id}/edit")
public String workProgramPostEdit(@PathVariable("id") Long id, @RequestParam int
number, @RequestParam int morenumber, @RequestParam String title, @RequestParam
int batch, @RequestParam String process,
                                  @RequestParam BigInteger machines, @RequestParam
String capacity, @RequestParam float onebatt_hour, @RequestParam String datetime,
Model model) {
    WorkProgram WorkProgram = WorkProgramRepository.findById(id).orElseThrow();
    Timestamp timestamp = Timestamp.valueOf(datetime);
    WorkProgram.setNumber(number);
    WorkProgram.setMorenumber(morenumber);
    WorkProgram.setTitle(title);
    WorkProgram.setBatch(batch);
    WorkProgram.setProcess(process);
    WorkProgram.setMachines(machines);
    WorkProgram.setCapacity(capacity);
    WorkProgram.setOnebatt_hour(onebatt_hour);
    WorkProgram.setDatetime(timestamp);
    WorkProgramRepository.save(WorkProgram);
    return "redirect:/base-plan";}
@PostMapping("/program/{id}/remove")
public String workProgramPostDelete(@PathVariable("id") Long id, Model model) {
    WorkProgram WorkProgram = WorkProgramRepository.findById(id).orElseThrow();
    WorkProgramRepository.delete(WorkProgram);
    return "redirect:/base-plan";
}
```

Рисунок 4.10, аркуш 2

4.5.3 Код контролеру ReportsController

Даний контролер створений для відображення усіх створених раніше робочих програм, але згрупованих за параметром моделі number. Тобто, програми з однаковим значенням цього параметру групуються в один блок (рис. 4.11).

Нижче наведено код для кнопки, при натисканні ня яку з’являється детальна інформація про визначену робочу програму, тому що використано метод findById. Виводиться детальна інформація про модель машини для виконання цієї програми, про дату початку виконання програми, за допомогою формули розраховується

значення параметру для тривалості виконання даної програми для визначеної кількості акумуляторів (згідно з формулами (2.1) – (2.2)). Далі результат перетворюється у час в форматі години : хвилини : секунди за допомогою утиліти TimeUtils (рис. 4.12)

```
@GetMapping("/reports/{id}/pt")
public String getPersonalTask(@PathVariable("id") Long id, Model model) {
    Optional<WorkProgram> optionalWorkProgram =
WorkProgramRepository.findById(id);
    if (optionalWorkProgram.isPresent()) {
        WorkProgram workProgram = optionalWorkProgram.get();
        BigInteger machineId = workProgram.getMachines();
Optional<Machines> optionalMachine =
machinesRepository.findById(machineId);
        double oneHourBatch = workProgram.getOnebatt_hour() *
workProgram.getBatch();
        double result;
        if ("MixerP".equals(workProgram.getTitle()) ||
"MixerE".equals(workProgram.getTitle())) {
            result = oneHourBatch * 0.0083;
        } else {
            result = oneHourBatch;
        }
        String formattedTime = TimeUtils.formatTime(result);
        model.addAttribute("formattedTime", formattedTime);
        if (optionalMachine.isPresent()) {
            Machines machine = optionalMachine.get();
            model.addAttribute("workProgram", workProgram);
            model.addAttribute("machine", machine);
            return "personal-tasks";
        }
    }
        return "redirect:/reports";
}
```

Рисунок 4.11 – Код контролеру ReportsController

```
public class TimeUtils {
    public static String formatTime(double result) {
        int hours = (int) Math.floor(result);
        int minutes = (int) Math.floor((result % 1) * 60);
        int seconds = (int) Math.floor((result % 1) * 3600 % 60);
        return String.format("%02d:%02d:%02d", hours, minutes, seconds);
    }
}
```

Рисунок 4.12 – Код утиліти TimeUtils

4.5.4 Код контролеру DispProcessesController

Даний контролер створений для відображення всіх робочих програм, виконання яких повинно початися у поточний день (рис. 4.13). Для цього в коді визначено початковий параметр startOfDay, що починає відлік з півночі поточного дня, та кінцевий endOfDay, для якого прописано значення 23,59,59. Після цього відібрано з переліку програм ті, які попадають в даний інтервал. Нижче реалізовано код при кліку на кнопку, який був описаний для контролеру ReportsController.

```
@GetMapping("/disp-processes")
public String showProcesses(Model model) {
    LocalDate currentDate = LocalDate.now();
    Timestamp startOfDay = Timestamp.valueOf(currentDate.atStartOfDay());
    Timestamp endOfDay = Timestamp.valueOf(currentDate.atTime(23, 59, 59))

    Iterable<WorkProgram> wprog =
workProgramRepository.findByDatetimeBetween(startOfDay, endOfDay);
    model.addAttribute("wprog", wprog);

    return "disp-processes";
}
```

Рисунок 4.13 – Код контролеру DispProcessesController

4.5.5 Код контролеру MonitoringController

Даний контролер створений для того, щоб передавати інформацію про усі активні машини підприємства (які знаходяться в статусі USED) за допомогою метода репозиторія кожної машини findUsedMachines на сторінку, а також передавати дані про програми на поточний день (рис. 4.14). Нижче наведений код для натиску на кнопку, при якому на сторінку repair завантажується список із активних машин таблиці machines. На цій сторінці диспетчер створює рапорт на ремонт та статус рапорту встановлюється на SENT. Після внесення змін у рапорт бригадиром ремонтної бригади статус змінюється на COMPLETED:

```
@GetMapping("/monitoring")
public String showMonitoring (Model model) {

    List<Object[]> findMachinesP = mixerpastaRepository.findUsedMachines();
    model.addAttribute("mixers", findMachinesP);

    List<Object[]> findMachinesE = mixerelectrolyteRepository.findUsedMachinesE();
    model.addAttribute("mixere", findMachinesE);}

@GetMapping("/monitoring/repair")
public String showRepairPage(Model model) {
    List<Machines> activeModels = MachinesRepository.findActiveModels();
    model.addAttribute("machines", activeModels);
    return "repair";
}
```

Рисунок 4.14 – Код контролеру MonitoringController

4.5.6 Код контролеру ReplaceMachinesController

Даний контролер створений для того, щоб передавати інформацію про усі машині на підприємстві для подальшої зміни їх статусів бригадиром ремонтної бригади (рис. 4.15). Усі машини підтягуються у контролер за допомогою методу findMachines. Далі для кожної машини виконується код, який приймає запит від користувача на оновлення статусу об’єкта, ідентифікуючи його по id, встановлює новий статус та зберігає зміни у базі даних.

```
@GetMapping("/replace-mach")
public String showReplaceMach(Model model) {
    List<Object[]> mixerpastaList = mixerpastaRepository.findMachines();
    model.addAttribute("mixerpastaList", mixerpastaList);}
@PostMapping("/update-status")
@ResponseBody
public String updateStatus(@RequestParam("id") BigInteger id,
@RequestParam("status") String status) {
            Mixerpasta mixerpasta =
mixerpastaRepository.findById(id).orElseThrow(() -> new
RuntimeException("Mixerpasta not found"));
            mixerpasta.setStatus(Mixerpasta.Status.valueOf(status));
mixerpastaRepository.save(mixerpasta);
            return "Status updated successfully";
}
```

Рисунок 4.15 – Фрагмент коду контролера ReplaceMachinesController для машини mixerpasta

4.5.7 Код контролеру DailyReportController

Даний контролер створений для виведення звітної інформації про виконані програми на поточний день та про ремонт, який було завершено (він у статусі COMPLETED). Дані про ремонт саме цього статусу виводяться за допомогою метода репозиторія findCompletedWithMachineModels (рис. 4.16).

```
@GetMapping("/daily-report")
public String showDailyReport(Model model) {
    LocalDate currentDate = LocalDate.now();
    Timestamp startOfDay = Timestamp.valueOf(currentDate.atStartOfDay());
    Timestamp endOfDay = Timestamp.valueOf(currentDate.atTime(23, 59, 59));

    List<Object[]> completedRepairsWithModels =
repairRepository.findCompletedWithMachineModels();
    model.addAttribute("repdaily", completedRepairsWithModels);
    Iterable<WorkProgram> workdaily =
workProgramRepository.findByDatetimeBetween(startOfDay, endOfDay);
    model.addAttribute("workdaily", workdaily);

    return "daily-report";
}
```

Рисунок 4.16 – Код контролера DailyReportController

4.5.8 Код репозиторію MixerpastaRepository

Даний репозиторій описує CRUD-запити до моделі mixerpasta (рис. 4.17). В ньому знаходиться два запити: перший за допомогою оператора SELECT збирає всі колонки з таблиці mixerpasta та за допомогою оператора JOIN замість числового значення зовнішнього ключа таблиці machines у колонці model відображає строкове значення колонки model таблиці machines, та надає машини тільки статусу USED. Таким чином, метод містить міксери статусу USED з назвами їх моделей замість зовнішнього ключа іншої таблиці. Другий метод подібний до першого, але відмінність в тому, що на метод поступають машини всіх статусів. Код репозиторію подібний до цього у інших машин та у репозиторія RepairRepository, тільки даний репозиторій у першому методі виводить рапорти зі статусом COMPLETED.

```
public interface MixerpastaRepository extends CrudRepository<Mixerpasta,
BigInteger> {

    @Query("SELECT mi.id, mi.speed_of_rot_min, mi.power_kwt, mi.noise_level_db,
mi.status, m.model " +
            "FROM Mixerpasta mi " +
            "JOIN Machines m ON mi.model = m.id " +
            "WHERE mi.status = 'USED'")
    List<Object[]> findUsedMachines();

    @Query("SELECT mi.id, mi.speed_of_rot_min, mi.power_kwt, mi.noise_level_db,
mi.status, m.model " +
            "FROM Mixerpasta mi " +
            "JOIN Machines m ON mi.model = m.id ")
    List<Object[]> findMachines();
}
```

Рисунок 4.17 – Код репозиторію MixerpastaRepository

4.5.9 Код репозиторію WorkProgramRepository

Даний репозиторій створений для виконання CRUD-запитів до таблиці work_program (рис. 4.18). В ньому реалізований метод findByDatetimeBetween, який виводить робочі програми у заданих діапазонах.

```
public interface WorkProgramRepository extends CrudRepository<WorkProgram, Long> {
    List<WorkProgram> findByDatetimeBetween(Timestamp start, Timestamp end);
}
```

Рисунок 4.18 – Код репозиторію WorkProgramRepository

4.5.10 Розробка тригерів

При зміні статусу машини (вибір між USED та STORAGE) бригадиром ремонтної бригади активна машина на підприємстві повинна змінитися не лише для нього та диспетчера, а й для проєктувальника. Через те, що проєктувальник працює лише з таблицею machines (її опис наведено у пункті 3.2.11), було вирішено додати даній таблиці колонку is_active, яка може містити або 0, або 1. Виходячи з цього було розроблено тригери для кожної з 9 таблиць машин підприємства, які

зв’язувалися з таблицею machines та при зміні статусу машини в таблиці machines змінювалось значення is_active. Якщо машина ставала USED – то значення змінювалося на 1, і навпаки. Приклад тригеру для таблиці mixerpasta наведено на рисунку 4.19.

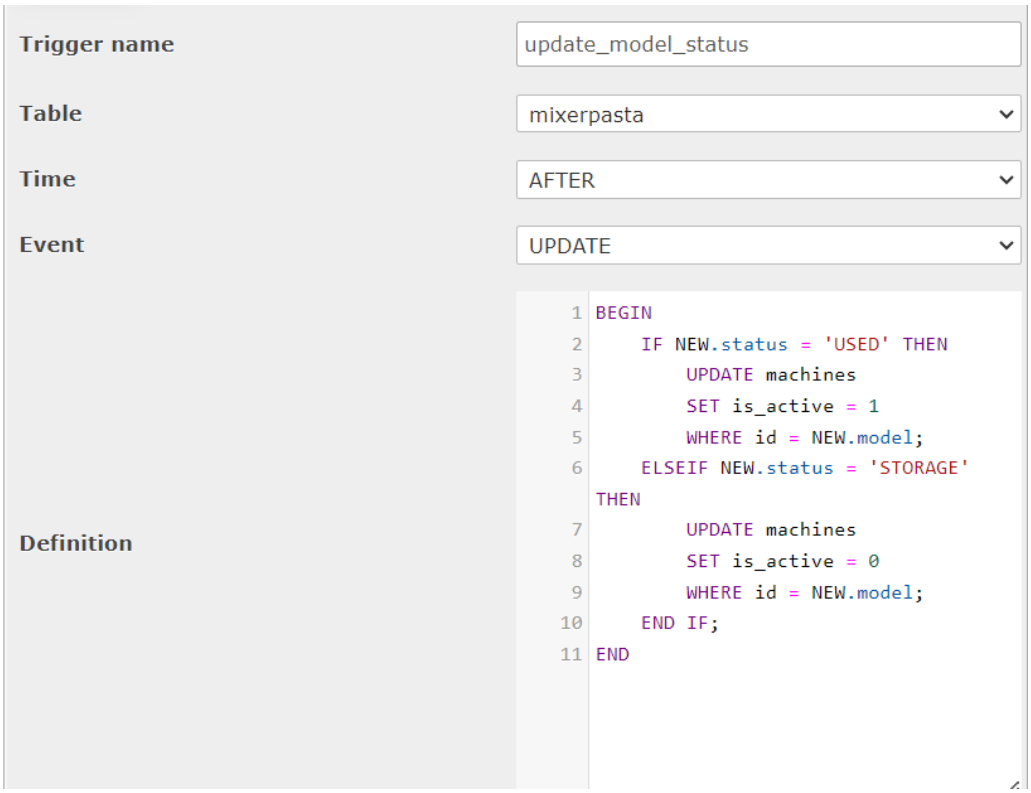

Рисунок 4.19 – Розроблений тригер для зміну статусу is_active

4.6 Розробка клієнтської частини системи

Клієнтська частина системи включає в себе HTML-файли з написаними всередені них стилями CSS та функціоналом-скриптами JavaScript. Для злагодженої роботи файлів було використано фреймворк Bootstrap, шаблонизатор Thymeleaf (Java-бібліотека, яка являє собою механізм для генерації вебсторінок на стороні сервера), бібліотеку Chart.js для побудови графіків. В розділах опису файлів були

використані лише деякі HTML, функціонал і код яких значно відрізняється від інших за будовою.

4.6.1 HTML-файл header

Даний файл зберігає в собі інформацію про шапку сайту для користувача Проєктувальник (рис. 4.20). В ній описана назва підприємства та вкладки, які доступні даному користувачеві. У диспетчера та бригадира схожий інтерфейс шапки, за виключенням інших назв вкладок.

```
<meta charset="UTF-8">
<div th:fragment="header">
<div class="container">
    <header class="d-flex flex-wrap justify-content-center py-3 mb-5 border-
bottom">
        <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto link-
body-emphasis text-decoration-none">
            <span class="fs-4">Telephone Maker</span>
        </a>

        <ul class="nav nav-pills">


            <li class="nav-item"><a href="/" class="nav-link active" aria-
current="page">Orders</a></li>
            <li class="nav-item"><a href="/program" class="nav-
link">Programs</a></li>
            <li class="nav-item"><a href="/base-plan" class="nav-link">Basic
plan</a></li>
            <li class="nav-item"><a href="/reports" class="nav-link"> Created
work-reports</a></li>
            <li class="nav-item"><a href="/daily-report" class="nav-link">Daily
report</a></li>
        </ul>
    </header>
</div>
</div>
```

Рисунок 4.20 – Код файлу header

4.6.2 HTML-файл work-prog

Даний файл створений для внесення нових робочих програм проєктувальником до бази даних (рис. 4.21). В ньому описані поля для вводу даних, випадаючі списки з даними та відображення вікна для вибору дати та часу. Також при виборі потрібного типу машини в полі onebatt_hour підставляється числове значення з таблиці machines з відповідного поля, це реалізовано за допомогою функції Javascript. Всі поля для вводу загорнуті у форму, яка має метод post, тому при натисканні на кнопку Add work-program нова програма додається до таблиці бази даних work_program. HTML-сторінка для редагування програми має таку ж структуру, але в ній до форми ще доданий об’єкт workProgram, описаний у контролері, щоб старі значення відображалися в полях при завантаженні сторінки.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Production program</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.c
ss">
    <link rel="stylesheet" href="/CSS/headercol.css">
    <style>
        .table td, .table th {
            background-color: #e6eaec;
        }
    </style>
    <script th:inline="javascript">
        function updateSize(select) {
            var selectedIndex = select.selectedIndex;
            var onebatt_hour =
select.options[selectedIndex].getAttribute('data-proconebKgH');
            document.getElementsByName('onebatt_hour')[0].value =
onebatt_hour;
        }
    </script>
</head>
```

Рисунок 4.21 – Код файлу work-prog

```
<body>
<header th:insert="blocks/header :: header"></header>
<div class="container mt-5">
    <h2>Production program</h2>
    <form action="/program" method="post">
        <div class="col-md-6 row mb-3 mt-3">
            <div class="col">
                <input type="text" id="datetimepicker" name="datetime"
placeholder="Enter begin date" class="form-control">
            </div>
            <div class="col">
                <input type="text" name="number" placeholder="Enter id-
number" class="form-control">
            </div>
            <div class="col">
                <input type="text" name="morenumber" placeholder="Enter
batch identifier" class="form-control">
            </div>
        </div>
        <div class="row mb-3">
            <div class="col">
                <select name="title" class="form-select">
                    <option value="" selected disabled>Select
title</option>
                    <option value="MixerP">MixerP</option>
                    <option value="MixerE">MixerE</option>
                    <option value="SyringeM">SyringeM</option>
                    <option value="DryOven">DryOven</option>
                    <option value="HydrP">HydrP</option>
                    <option value="LaserC">LaserC</option>
                    <option value="StackMash">StackMash</option>
                    <option
value="CellElectrolyte">CellElectrolyte</option>
                    <option value="SealCham">SealCham</option>
                </select>
            </div>
            <div class="col">
                <input type="text" name="batch" placeholder="Enter number
of batch" class="form-control">
            </div>
            <div class="col">
                <select name="capacity" class="form-select">



             <div class="col">
                 <select name="capacity" class="form-select">
                     <option value="" selected disabled>Select
 capacity</option>
                     <option value="6000">6000</option>
                     <option value="5000">5000</option>
                     <option value="5500">5500</option>
                 </select>
             </div>
         </div>
         <table class="table">
             <thead>
             <tr>
```

Рисунок 4.21, аркуш 2

```
                        <th>Process</th>
                    <th>Machines</th>
                    <th>Processing time per battery in hours</th>
                </tr>
                </thead>
                <tbody>
                <tr>
                    <td>
                        <select name="process" class="form-select">
                            <option value="" selected disabled>Select
process</option>
                            <option value="mixing_paste">Making paste</option>
                            <option value="creating_electrolyte">Making
electrolyte</option>
                            <option
value="applying_paste_to_collectors">Applying paste to collectors</option>
                            <option value="drying_the_electrodes">Drying the
electrodes</option>
                            <option value="electrode_pressing">Electrode
pressing</option>
                            <option value="electrode_cutting">Cutting
electrodes</option>
                            <option value="stacking_process">Stacking
process</option>
                            <option value="filling_electrolyte">Filling the
battery with electrolyte</option>
                            <option value="sealing_of_battery_cells">Sealing of
battery cells</option>
                        </select>
                    </td>
                    <td>
                        <select name="machines" class="form-select"
onchange="updateSize(this)">
                            <option value="" selected disabled>Select
machine</option>
                            <option th:each="machines : ${machines}"
th:value="${machines.id}" th:text="${machines.type}" th:attr="data-
proconebKgH=${machines.proconeb_kg_h}"></option>
                        </select>
                    </td>
                    <td><input type="text" name="onebatt_hour"
placeholder="Enter processing time" class="form-control" readonly></td>


                </tr>
                </tbody>
            </table>
            <input type="hidden" th:name="${_csrf.parameterName}"
th:value="${_csrf.token}" />
            <button type="submit" class="btn btn-warning text-black">Add work-
program</button>
        </form>
</div>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jquery-
datetimepicker/2.5.20/jquery.datetimepicker.min.css">
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

Рисунок 4.21, аркуш 3

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
datetimepicker/2.5.20/jquery.datetimepicker.full.min.js"></script>
<script>
    $(function() {
        var allowedTimes = [];
        for (var hour = 8; hour <= 15; hour++) {
            for (var minute = 0; minute < 60; minute++) {
                allowedTimes.push(String(hour).padStart(2, '0') + ':' +
String(minute).padStart(2, '0'));
            }
        }
        $("#datetimepicker").datetimepicker({
            format: 'Y-m-d H:i:s',
            step: 1,
            allowTimes: allowedTimes }); });
</script>
</body>
</html>
```

Рисунок 4.21, аркуш 4

4.6.3 HTML-файл base-plan

Даний файл створений для виведення всіх створених програм (рис. 4.22). Вони групуються за своєю назвою у окремі блоки, на яких реалізована функція toggleBlock, яка при кліку на кнопку надає можливість ховати/відкривати вміст блоків. У кожної програми реалізовані її власні кнопки редагування (при кліку веде на сторінку редагування програми) та видалення (діє функція delete).

Файл має дві вкладки на самій сторінці: на одній відображаються блоки з вмістом програм, на іншій графік, побудований за допомогою бібліотеки Chart.js. Для кожного блоку з програмами свій окремий графік. Щоб його побачити, необхідно клікнути на блок з програмами та перейти на вкладку з графіком. Ось X залежить від дати початку роботи програми, а для осі Y (на ній відображається, наскільки машина завантажена у цей день) розраховується значення параметру для тривалості виконання програми для визначеної кількості акумуляторів (за формулами (2.1) – (2.2)).

Якщо дві програми мають однакову дату початку роботи – то їх значення тривалості виконання програми сумуються і відображаються однією лінією. Якщо

значення завантаженості програми перевищує коефіцієнт 6 (тобто машина працює більше 6 годин з 8 робочих), то ця лінія відображається червоним кольором і це свідчить про те, що проєктувальнику слід або перенести програму на інший день, або розбити її на дві і переносити частинами. Фрагмент наведений для блоку mixerpasta та JavaScript-функціоналу усіх блоків.

```
<div class="container mt-5">
    <ul class="nav nav-tabs" id="myTabs" role="tablist">
        <li class="nav-item" role="presentation">
            <button class="nav-link active" id="blocks-tab" data-bs-toggle="tab"
data-bs-target="#blocks" type="button"
                    role="tab" aria-controls="blocks" aria-selected="true">Basic
plan of programs
            </button></li>
        <li class="nav-item" role="presentation">
            <button class="nav-link" id="chart-tab" data-bs-toggle="tab" data-bs-
target="#chart" type="button"
                    role="tab" aria-controls="chart" aria-selected="false">Diagram
            </button></li></ul>
    <div class="tab-content" id="myTabContent">
        <div class="tab-pane fade show active" id="blocks" role="tabpanel" aria-
labelledby="blocks-tab">
            <h2>Basic plan of programs</h2>
            <div th:with="mixerPElements=${progs.?[title == 'MixerP']},
                  mixerEElements=${progs.?[title == 'MixerE']},
                  syringeElements=${progs.?[title == 'SyringeM']},
                  dryovenElements=${progs.?[title == 'DryOven']},
                  hydrpElements=${progs.?[title == 'HydrP']},
                  lasercElements=${progs.?[title == 'LaserC']},
```

Рисунок 4.22 – Фрагменти коду файлу base-plan

```
                stackmashElements=${progs.?[title == 'StackMash']},
                cellelectrolyteElements=${progs.?[title == 'CellElectrolyte']},
                sealchamElements=${progs.?[title == 'SealCham']}">

        <div th:if="${not #lists.isEmpty(mixerPElements)}" class="mixerp-block">
                <h4>
                    MixerP Blocks
                    <button class="btn btn-outline-light float-end"
onclick="toggleBlock(this)">Hide/Show</button>
                </h4>
                <div class="mixerp-block-body hidden">
                    <div th:each="el : ${mixerPElements}" class="alert alert-
info mt-2">
                        <h3 th:text="'Id: ' + ${el.title} + '-' + ${el.number}
+ ((${el.morenumber} != null) ? ('-' + ${el.morenumber}) : '')"></h3>
                        <p><strong>Title:</strong> <span
th:text="${el.title}"></span></p>
                        <p><strong>Beginning of work:</strong> <span
th:text="${el.datetime}"></span></p>
                        <p><strong>Batch:</strong> <span
th:text="${el.batch}"></span></p>
                        <p><strong>Capacity:</strong> <span
th:text="${el.capacity}"></span></p>
                        <p><strong>Process:</strong> <span
th:text="${el.process}"></span></p>
                        <p><strong>Machines:</strong> <span
th:text="${el.machines}"></span></p>
                        <p><strong>Processing time for one battery in
kg/hours:</strong> <span th:text="${el.onebatt_hour}"></span></p>
                        <div class="button-container">
                            <a th:href="@{'/program/' + ${el.id} + '/edit'}"
class="btn btn-outline-success">Update</a>
                            <form th:action="'/program/' + ${el.id} +
'/remove'" method="post" style="display: inline-block;">
                                <button class="btn btn-outline-danger"
type="submit">Delete</button></form></div></div></div></div>
<div class="tab-pane fade" id="chart" role="tabpanel" aria-labelledby="chart-tab">
    <!-- Контейнер для графіка -->
    <div>
        <canvas id="myChart"></canvas>
    </div></div><script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

Рисунок 4.22, аркуш 2

```
<script>
    const myChart = document.getElementById('myChart');
    let chart
    const blockElements = document.querySelectorAll('.mixerp-block,
.mixere-block, .syringem-block, .dryoven-block, .hydrp-block, .laserc-
block, .stackmash-block, .cellelectrolyte-block, .sealcham-block');
    blockElements.forEach(block => {
        block.addEventListener('click', () => {
            const blockTitle =
block.querySelector('h4').textContent.trim().split(' ')[0];
const elements = block.querySelectorAll('.alert-info');
            if (blockTitle === 'MixerP') {
                                const data = processMixerPData(elements);
                createOrUpdateChart(data);
            } else {
                const data = {
                    labels: Array.from(elements).map(el =>
el.querySelector('p:nth-child(3) span').textContent) };
                createOrUpdateChart(data); } }); });
    function processMixerPAndEData(elements, blockTitle) {
        const labels = [];
        const data = [];
        const groupedByDatetime = Array.from(elements).reduce((acc, el) =>
{
            const datetimeStr = el.querySelector('p:nth-child(3)
span').textContent;
            const datetime = datetimeStr.split(' ')[0];
            const title = el.querySelector('p:nth-child(2)
span').textContent;
            if (!acc[datetime]) {
                acc[datetime] = [];
            }
            if (!acc[datetime][title]) {
                acc[datetime][title] = [];
            }
            acc[datetime][title].push(el);
            return acc;
        }, {});
        for (const [datetime, groupByTitle] of
Object.entries(groupedByDatetime)) {
            let sum = 0;
            for (const [title, group] of Object.entries(groupByTitle)) {


                for (const el of group) {
                    const batch = parseFloat(el.querySelector('p:nth-
    child(4) span').textContent);
                    const onebattHour = parseFloat(el.querySelector('p:nth-
    child(8) span').textContent);
                    sum += 0.0083 * (onebattHour * batch);
                }
            }
            labels.push(datetime);
            data.push(sum);
        }
        return { labels, datasets: [{ label: blockTitle, data }] };
    }
     function processOtherData(elements, title) {
        const labels = [];
        const data = [];
        const groupedByDatetime = Array.from(elements).reduce((acc, el) =>
    {
            const datetimeStr = el.querySelector('p:nth-child(3)
    span').textContent;
            const datetime = datetimeStr.split(' ')[0];
            if (!acc[datetime]) {
                acc[datetime] = [];
            }
            acc[datetime].push(el);
```

Рисунок 4.22, аркуш 3

```
<script>
    const myChart = document.getElementById('myChart');
    let chart;
    const blockElements = document.querySelectorAll('.mixerp-block, .mixere-block,
.syringem-block, .dryoven-block, .hydrp-block, .laserc-block, .stackmash-block,
.cellelectrolyte-block, .sealcham-block');
    blockElements.forEach(block => {
        block.addEventListener('click', () => {
            const blockTitle =
block.querySelector('h4').textContent.trim().split(' ')[0];


            const elements = block.querySelectorAll('.alert-info');
            if (blockTitle === 'MixerP') {
                const data = processMixerPData(elements);
                createOrUpdateChart(data);
            } else {
                const data = {
                    labels: Array.from(elements).map(el =>
  el.querySelector('p:nth-child(3) span').textContent)};
                createOrUpdateChart(data);} }); });
    function processMixerPAndEData(elements, blockTitle) {
        const labels = [];
        const data = [];
        const groupedByDatetime = Array.from(elements).reduce((acc, el) => {const
  datetimeStr = el.querySelector('p:nth-child(3) span').textContent;
            const datetime = datetimeStr.split(' ')[0]; "
            const title = el.querySelector('p:nth-child(2) span').textContent;
            if (!acc[datetime]) {
                acc[datetime] = [];}
            if (!acc[datetime][title]) {
                acc[datetime][title] = [];}
            acc[datetime][title].push(el);
            return acc;}, {});
        for (const [datetime, groupByTitle] of Object.entries(groupedByDatetime))
  {
            let sum = 0;
            for (const [title, group] of Object.entries(groupByTitle)) {
                for (const el of group) {
                    const batch = parseFloat(el.querySelector('p:nth-child(4)
  span').textContent);
                    const onebattHour = parseFloat(el.querySelector('p:nth-
  child(8) span').textContent);
                    sum += 0.5 * (onebattHour * batch); } }
            labels.push(datetime);
            data.push(sum); }
        return { labels, datasets: [{ label: blockTitle, data }] }; }
    function processOtherData(elements, title) {
        const labels = [];
        const data = [];
        const groupedByDatetime = Array.from(elements).reduce((acc, el) => {const
  datetimeStr = el.querySelector('p:nth-child(3) span').textContent;
            const datetime = datetimeStr.split(' ')[0];
            if (!acc[datetime]) {
                acc[datetime] = [];}
            acc[datetime].push(el);
```

Рисунок 4.22, аркуш 4

```
                        return acc;}, {});
        for (const [datetime, group] of Object.entries(groupedByDatetime)) {let
sum = 0;
            for (const el of group) {
                const batch = parseFloat(el.querySelector('p:nth-child(4)
span').textContent);
                const onebattHour = parseFloat(el.querySelector('p:nth-child(8)
span').textContent);
                sum += onebattHour * batch; }
            labels.push(datetime);
            data.push(sum);
        } return { labels, datasets: [{ label: title, data }] };
    } function createOrUpdateChart(data) { if (chart) {
                        chart.data.labels = data.labels;
```

```
                                    chart.data.datasets =
data.datasets.map(dataset => {
                const colors = dataset.data.map(value => value > 6 ? 'rgb(255, 99,
132)' : 'rgb(54, 162, 235)');
                return {
                    ...dataset,
                    backgroundColor: colors }; });
            chart.update();
        } else {
            chart = new Chart(myChart, {
                type: 'bar',
                data: {
                    labels: data.labels,
                    datasets: data.datasets.map(dataset => {
                        const colors = dataset.data.map(value => value > 6 ?
'rgb(255, 99, 132)' : 'rgb(54, 162, 235)');
                        return {
                            ...dataset,
                            backgroundColor: colors }; }) }, options: {
                    indexAxis: 'y',
                    scales: {
                        x:beginAtZero: true,} } } }); } }
    let selectedBlock = null;
    function highlightBlock(block) {
        if (selectedBlock) {
            selectedBlock.style.backgroundColor = '';
            selectedBlock.style.color = ''; }
        if (block) {
            block.style.backgroundColor = '#5253f9';
            block.style.color = '#b5dbfc'; }
        selectedBlock = block;
    } blockElements.forEach(block => {
        block.addEventListener('click', () => {
            const blockTitle =
block.querySelector('h4').textContent.trim().split(' ')[0];
            const elements = block.querySelectorAll('.alert-info');
            if (blockTitle === 'MixerP' || blockTitle === 'MixerE') {
                // Спеціальна обробка для блоків MixerP и MixerE
                const data = processMixerPAndEData(elements, blockTitle);
                createOrUpdateChart(data);
                highlightBlock(block);
            } else {
                const data = processOtherData(elements, blockTitle);
                createOrUpdateChart(data);
```

Рисунок 4.22, аркуш 5

```
            highlightBlock(block); } }); });
    function toggleBlock(button) {
        const blockBody = button.parentElement.nextElementSibling;
        blockBody.classList.toggle('hidden');
        button.innerText = blockBody.classList.contains('hidden') ? 'Show' :
'Hide'; }
</script>
<script>
    document.getElementById('blocks-tab').addEventListener('click', function () {
location.reload(); });
</script>
```

Рисунок 4.22, аркуш 6

4.6.4 HTML-файл reports

В даному файлі реалізовано виведення всіх робочих програм, згрупованих за параметром number, в табличному вигляді (рис. 4.23). Також на даній сторінці реалізовано функцію виведення на друк за допомогою кнопки, при якому дана кнопка та шапка сайту приховується і залишається лише основна частина, виділена розробником в коді як printContent.

```
<h2>Reports</h2>
<button class="btn btn-outline-primary" onclick="printEl()"><i class="bi bi-
printer"></i> Print</button>
</div>
<div id="programBlocks">
    <th:block th:each="entry : ${groupedPrograms}">
        <div class="card mb-3">
            <div class="card-header">
                <h5 class="card-title" th:text="'Program №' + ${entry.key}"></h5>
            </div>
            <div class="card-body">
                <table class="table"></table></div></div></th:block></div></div>
</div><script>
    function printEl(){
        var printButton = document.querySelector('.btn.btn-outline-primary');
        printButton.style.display = 'none';
        var printWindow = window.open('', '_blank');
        var printContent = document.getElementById('printContent').innerHTML;
        var styles = document.getElementsByTagName('link');
        var stylesHTML = '';
        for (var i = 0; i < styles.length; i++) {
            stylesHTML += styles[i].outerHTML;}
        printWindow.document.write('<html><head><title>Print</title>');
        printWindow.document.write(stylesHTML);
        printWindow.document.write('</head><body>');
        printWindow.document.write(printContent);
        printWindow.document.write('</body></html>');
        printWindow.document.close();
        printWindow.print();
        printWindow.close();
        printButton.style.display = ''; }
</script>
```

Рисунок 4.23 – Фрагмент коду функції друку та групування

4.6.5 HTML-файл monitoring

Даний файл описує функціонал для моніторингу процесів виробництва диспетчером (рис. 4.24). В ньому є кнопки для запуску процесу, його зупинки та кнопка для написання рапорту на виклик ремонтної бригади. Нижче зображена таблиця з моделями машин та їх основними показниками. Нижче наведено 9 таблиць для кожної машини, в яких більш детальні параметри їх роботи. При запуску системи починається генерація чисел у 9 таблицях для кожної машини.

Також після запуску системи на екрані дисплея диспетчера з’являються повідомлення-тости про завершення тої чи іншої програми із запланованих на поточний день. При виникненні неполадки з’являється повідомлення з червоним текстом, та на верхній таблиці відображається значення Abnormal у потрібній колонці. Нижче, у таблицях з детальними параметрами, можна побачити числове значення неполадки. Після цього диспетчер зупиняє робочі процеси та переходить на створення рапорту про ремонт несправності. У фрагментах коду наведено код для рядка таблиці mixerpasta, код для таблиці mixerpasta, генерація чисел для даної таблиці та виклики помилок і тостів. Для інших таблиць схожа реалізація, тому їх код не приведено.

```
<a th:href="@{'/monitoring/repair'}" class="btn btn-primary me-4">
        <i class="bi bi-people"></i> Repair</a></div>
<div class="tab-content" id="myTabContent">
        <div class="row">
            <div class="col-md-12">
                <table class="table">
                    <thead><tr>
                        <th>Model</th>
                        <th>Type</th>
                        <th>Productivity</th>
                        <th>Downtime</th>
                        <th>Noise generation</th>
                    </tr></thead><tbody>
                    <tr th:each="mixer : ${mixers}">
                        <td th:text="${mixer[5]}"></td>
                        <td>Mixer for pasta</td>
                        <td>
                            <span class="normal">Normal</span></td>
                        <td>
```

Рисунок 4.24 – Фрагменти коду файлу monitoring

```
<a th:href="@{'/monitoring/repair'}" class="btn btn-primary me-4">
        <i class="bi bi-people"></i> Repair</a></div>
<div class="tab-content" id="myTabContent">
        <div class="row">
            <div class="col-md-12">
                <table class="table">
                    <thead><tr>
                        <th>Model</th>
                        <th>Type</th>
                        <th>Productivity</th>
                        <th>Downtime</th>
                        <th>Noise generation</th>
                    </tr></thead><tbody>
                    <tr th:each="mixer : ${mixers}">
                        <td th:text="${mixer[5]}"></td>
                        <td>Mixer for pasta</td>
                        <td>
                            <span class="normal">Normal</span></td>
                        <td>
                            <span class="normal">Normal</span> <!-- Downtime -->
                        </td>
                        <td>
                         <span id="noiseGenerationStatus" class="normal">Normal</span>
                            </td></tr></tbody></table></div>
<div class="col-md-6">
    <div id="mixer-details" class="row">
        <div th:each="mixer, iterStat : ${mixers}" class="col-md-6">
            <table class="table">
                <thead>
                <tr>
                    <th>Model</th>
                    <th>Speed of rotation</th>
                    <th>Power (KWt)</th>
                    <th>Noise level (dB)</th>
                </tr></thead><tbody>
                <tr>
                    <td th:text="${mixer[5]}" data-model="${mixer[6]}"></td>
                    <td>-</td>
                    <td th:text="${mixer[2]}"></td>
                    <td>-</td>
                </tr></tbody></table></div></div>
let noiseGenerationStatusElement =
document.getElementById('noiseGenerationStatus');
setTimeout(function() {
    noiseGenerationStatusElement.textContent = 'Abnormal';


   noiseGenerationStatusElement.classList.remove('normal');
   noiseGenerationStatusElement.classList.add('abnormal');
   showToast('Increased noise level detected', true);
}, 180000);
function startProcesses() {
    // Генерація випадкових чисел для таблиць міксерів
    let mixerSpeedCells = document.querySelectorAll('#mixer-details table tbody tr
td:nth-child(2)');
    let mixerNoiseCells = document.querySelectorAll('#mixer-details table tbody tr
td:nth-child(4)');
let startTime = new Date().getTime();
function updateMixerValues() {
    mixerSpeedCells.forEach(cell => {
        cell.textContent = generateRandomNumber(60, 70); });
```

Рисунок 4.24, аркуш 2

```
mixerNoiseCells.forEach(cell => {
    cell.textContent = generateRandomNumber(78, 80);});
let elapsedTime = new Date().getTime() - startTime;
            if (elapsedTime >= 180000) {
                mixerNoiseCells.forEach(cell => {
                    cell.textContent = 82;
                    cell.style.color = 'red';});
                clearInterval(intervalId); } }
        let intervalId = setInterval(updateMixerValues, 60000);
        updateMixerValues();
        processStartTime = new Date().getTime();
        showProcessToasts();
        showErrorToast(); }
    function generateRandomNumber(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;}
        let pauseButton = document.querySelector('.btn-primary:nth-child(2)');
        pauseButton.addEventListener('click', function() {
        if (!pauseFlag) {
            clearInterval(mixerIntervalId);
            processTimeouts.forEach(timeout => clearTimeout(timeout));
            pauseFlag = true;
        } else {
            mixerIntervalId = setInterval(updateMixerValues, 60000);
            pauseFlag = false;
            this.textContent = 'Pause'; }});
    function showProcessToasts() {
        clearInterval(processIntervalId);
        setTimeout(function() {
            const currentTime = new Date().getTime();
            const elapsedTime = currentTime - processStartTime;
                        for (let i = 0; i < processNames.length; i++) {
                    const delay = i * 120000;             setTimeout(function() {
        showToast(processNames[i], false);}, delay); }
            processIntervalId = setInterval(showProcessToasts, 86400000);
}, 120000);    }
    function showToast(text, isError = false) {
        const toast = isError ? document.getElementById("errorToast") :
document.getElementById("processToast");
        const messageElement = isError ? document.getElementById("errorMessage") :
document.getElementById("processName");
        messageElement.textContent = isError ? text : `Program "${text}" done
success`;
        toast.classList.add("show");
        setTimeout(function() {

           toast.classList.remove("show");
       }, isError ? 7000 : 3000);    }
   function showErrorToast() {
       setTimeout(function() {
           showToast('Increased noise level detected', true);}, 180000); }
   /*]]>*/</script>
```

Рисунок 4.24, аркуш 3

4.6.6 HTML-файл replace-mach

Даний файл описує можливість бригадира ремонтної бригади змінювати статуси машин (рис. 4.25). Статуси оновлюються для визначеного id на сервері за допомогою використання AJAX-запитів, без перезавантаження сторінки. Після успішного оновлення показується повідомлення-тост, яке говорить про успішне оновлення статусу. У фрагментах коду наведено код для машин mixerpasta та код AJAX-запиту до таблиці. Для інших таблиць схожа реалізація, тому їх код не приведено.

```
<div class="tab-content" id="myTabContent">
    <div class="row">
        <div class="col-md-12 mb-5">
            <h3>Mixers for pasta</h3>
            <table class="table table-striped">
                <thead>
                <tr>
                    <th>Model</th>
                    <th>Status</th>
                    <th>Action</th>
                </tr>
                </thead>
                <tbody>
                <tr th:each="item : ${mixerpastaList}">
                    <td th:text="${item[5]}"></td>
                    <td>
                        <select th:id="'status-' + ${item[0]}" class="form-
select">
                            <option value="USED" th:selected="${item[4].name() ==
'USED'}">USED</option>
                            <option value="STORAGE" th:selected="${item[4].name()
== 'STORAGE'}">STORAGE</option>
                        </select>
                    </td>
                    <td>
                        <button type="button" th:data-id="${item[0]}" class="btn
btn-primary save-status">Save changes</button>
                    </td>
                </tr>
                </tbody>
            </table>
        </div>
<script>
    $(document).ready(function() {
```

Рисунок 4.25 – Фрагменти коду файлу replace-mach

```
            </table>
        </div>
<script>
    $(document).ready(function() {
        $('.save-status').click(function() {
            var id = $(this).data('id');
            var status = $('#status-' + id + ' option:selected').val();

            $.ajax({
                url: '/update-status',
                type: 'POST',
                data: {
                    id: id,
                    status: status,
                },
                success: function(response) {
                    var toast = new bootstrap.Toast($('#statusToast'));
                    toast.show();
                },
                error: function(xhr, status, error) {
                    console.error(error);
                }
            });
        });
    });
</script
```

Рисунок 4.25, аркуш 2

ВИСНОВКИ

В ході виконання роботи було проведено аналіз предметної області, визначено основні вимоги до розроблюваного компоненту MES-системи на виробництві. Були виконані наступні завдання:

– розглянуто концепцію MES-системи, її роль в оптимізації процесів та її основні існуючі компоненти;

– проведено аналіз основних проблем виробництва та доведено доцільність впровадження системи MES-для вирішення даних питань;

– визначено головну мету кваліфікаційної роботи, виділено основні системні вимоги до розроблюваної системи, спроєктовано роботу системи за допомогою створення діаграм мовою UML.

– розроблено табличну частину бази даних предметної області;

– проведено нормалізацію створеної бази даних;

– проведено обгрунтування вибору СУБД, мови програмування, допоміжних фреймворків, середи розробки та додатків для взаємодії з базою даних;

– проведено розробку серверної частини проєкту;

– проведено розробку клієнтської частини проєкту.

Розроблене програмне забезпечення може бути інтегрованим у виробництво для забезпечення його цілісності, мінімізації простоїв обладнання та підвищення ефективності роботи підприємства в цілому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. 29 с.

2. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» та 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» освітньої програми «Системна інженерія» / І. Ш. Невлюдов та ін. Харків : МОН України, ХНУРЕ, 2023. 218 с.

3. Дипломне проєктування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології»: навч. посібник / І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, Г.В. Пономарьова. Київ, 2018. 320 с.

4. Вінниченко С.О., Колесник Л.В. Інтеграція MES-системи в сучасні підприємства: переваги і недоліки // «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві», 22 листопада 2023. Харків, Україна. 2023. С. 131-134.

5. Вінниченко С.О. Еволюція виробництва: Роль MES-системи у оптимізації та контролі промислових процесів на підприємстві // Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2023) : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. Харків : ХНУРЕ, 2023. – Вип. 2. – 408с.

6. What is a Manufacturing Execution System (MES)? | IBM. *IBM in Deutschland, Österreich und der Schweiz*. URL: https://www.ibm.com/topics/mes-system (дата звернення: 08.05.2024).

7. Transforming semiconductor manufacturing at Littelfuse. *Cantier*. URL: https://www.cantier.com/case-studies/case-study-littelfuse-semiconductor/ (дата звернення: 08.05.2024).

8. Основи UML. *KDE Documentation* - URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (дата звернення: 11.06.2024).

9. Research on micromechanical behavior of current collector of lithium-ion batteries battery cathode during the calendering process. *MDPI*. URL: https://www.mdpi.com/2227-9717/11/6/1800 (дата звернення: 06.06.2024).

10. Post-Mortem analysis of inhomogeneous induced pressure on commercial lithium-ion pouch cells and their effects / G. Fuchs та ін. *Sustainability*. 2019. Т. 11, № 23. С. 6738. URL: https://doi.org/10.3390/su11236738 (дата звернення: 07.06.2024).

11. Different types of lithium polymer batteries. *Leading Provider of Custom Lithium Battery Pack Solution | Grepow*. URL: https://www.grepow.com/blog/different-types-of-lithium-polymer-batteries.html (дата звернення: 06.06.2024).

12. A comprehensive understanding of electrode thickness effects on the electrochemical performances of Li-ion battery cathodes / H. Zheng та ін. *Electrochimica acta*. 2018. Т. 71. С. 258–265.
URL: https://doi.org/10.1016/j.electacta.2012.03.161 (дата звернення: 07.06.2024).

13. 5 litre lab planetary vacuum mixer machine, machine suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/5-litre-lab-planetary-vacuum-mixer-machine_p796.html (дата звернення: 06.06.2024).

14. Electrolytes, additives and binders for NMC cathodes in li-ion batteries–a review. *MDPI*. URL: https://www.mdpi.com/2313-0105/9/4/193 (дата звернення: 06.06.2024).

15. lab small battery electrode film roller coater slot die coating machine manufacturer, *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/lab-small-battery-

electrode-film-roller-coater-slot-die-coating-machine-manufacturer_p988.html (дата звернення: 06.06.2024).

16. Optimized lifepo4-based cathode production for lithium-ion batteries through laser-and convection-based hybrid drying process. *MDPI*.
URL: https://www.mdpi.com/2032-6653/14/10/281 (дата звернення: 06.06.2024).

17. LiPF6 standard electrolyte for lithium-ion batteries by E-Lyte Innovations. *E-Lyte Innovations*. URL: https://e-lyte.de/products/electrolytes-for-batteries/lipf6-based-electrolytes/#:~:text=Among%20all%20conducting%20salts,%20lithium,current%20collector%20passivation%20and%20effective (дата звернення: 06.06.2024).

18. High temperature vacuum oven,53l 500℃ high temperature vacuum oven suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/53l-500-high-temperature-vacuum-oven_p536.html (дата звернення: 06.06.2024).

19. Roll heat press machine,hot press machine,lab calender machine. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/150-200mm-roll-heat-press-machine-for-battery-electrode_p179.html (дата звернення: 06.06.2024).

20. Semi automatic slitting machine for battery electrode,semi automatic slitting machine for battery electrode suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/semi-automatic-slitting-machine-for-battery-electrode_p278.html (дата звернення: 06.06.2024).

21. Theoretical analysis of potential and current distributions in planar electrodes of lithium-ion batteries / P. Taheri та ін. *Electrochimica acta*. 2017. Т. 133. С. 197–208. URL: https://doi.org/10.1016/j.electacta.2014.04.040 (дата звернення: 07.06.2024).

22. Automatic battery electrode stacking machine for lithium pouch cell,automatic battery electrode stacking machine for lithium pouch cell suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials*

Manufacturer. URL: https://www.tobmachine.com/automatic-battery-electrode-stacking-machine-for-lithium-pouch-cell_p1101.html (дата звернення: 06.06.2024).

23. Vacuum filling machine for cylindrical battery,vacuum filling machine for cylindrical battery suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/vacuum-filling-machine-for-cylindrical-battery_p102.html (дата звернення: 06.06.2024).

24. Numerical models of the electrolyte filling process of lithium-ion batteries to accelerate and improve the process and cell design. *MDPI*. URL: https://www.mdpi.com/2313-0105/8/10/159 (дата звернення: 06.06.2024).

25. Pouch battery electrolyte vacuum diffusion chamber with pre-sealing all-in-one machine,pouch battery electrolyte vacuum diffusion chamber with pre-sealing all-in-one machine suppliers. *Battery Machine,Supercapacitor Equipment,Battery Production Line And Battery Materials Manufacturer*. URL: https://www.tobmachine.com/pouch-battery-electrolyte-vacuum-diffusion-chamber-with-pre-sealing-all-in-one-machine_p860.html (дата звернення: 06.06.2024).

26. Wojtyto D., Michalik J., Angelova M. Noise level measurement and analysis in a manufacturing enterprise. *System safety: human - technical facility - environment*. 2021. Т. 3, № 1. С. 192–200. URL: https://doi.org/10.2478/czoto-2021-0020 (дата звернення: 12.06.2024).

27. Методичні вказівки до виконання розділу «Охорона праці» у випускних роботах ОКР «бакалавр» усіх форм навчання [Текст] / упоряд.: Б. В. Дзюнзюк, В. А. Айвазов, Т. Є. Стиценко. – Харків: ХНУРЕ, 2012. – 28 с

28. Що таке ER Modeling? Навчайтеся на прикладі. *Guru99*. URL: https://www.guru99.com/uk/er-modeling.html (дата звернення: 11.06.2024).

29. Бази даних. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів | BestProg. *BestProg | Програмування: теорія та практика*.

URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/ (дата звернення: 30.05.2024).

30. Реляційні бази даних. *Relational databases*. URL: https://rdb.dp.ua/uk/chapter_03 (дата звернення: 30.05.2024).

31. Contributors to Вікі Програмування для чайників. Java. *Вікі Програмування для чайників*. URL: https://newbieprogramming.fandom.com/uk/wiki/Java (дата звернення: 30.05.2024).

32. Що таке spring boot? Ввідна інформація про spring boot | microsoft azure. *Cloud Computing Services | Microsoft Azure*. URL: https://azure.microsoft.com/resources/cloud-computing-dictionary/what-is-java-spring-boot (дата звернення: 30.05.2024).

33. Керування базами даних та інтеграція з веб-додатками: MySQL, MongoDB і SQL Server. *RedStone*. URL: https://redstone.media/keruvannia-bazamy-danych-ta-intehracija-z-veb-dodatkamy-mysql-mongodb-i-sql-server (дата звернення: 30.05.2024).

34. Що таке hibernate в java?. *itProger*. URL: https://itproger.com/ua/spravka/java/hibernate (дата звернення: 30.05.2024).