Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно–інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення автоматичного програмного модуля адміністрування підприємством
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТСІ–21–3
Дмитро ЗАЛУЦЬКИЙ
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно–інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо–професійна
Освітня програма Автоматизація та комп'ютерно–інтегровані технології
(повна назва освітньої програми)

Керівник Дмитро ГУРІН
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Залуцький Дмитро Дмитрович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

"24" червня 2025 р.  Дмитро ЗАЛУЦЬКИЙ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно–інтегровані технології
(код і повна назва)
Тип програми Освітньо–професійна
Освітня програма Автоматизація та комп'ютерно–інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 03 » травня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Залуцькому Дмитру Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення автоматичного програмного модуля адміністрування підприємством

Затверджена наказом по університету від 19 травня 2025 р. №

2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2025 р.
3. Вихідні дані до роботи Мови програмування: PHP, JavaScript; СКБД – MySQL; тип інтерфейсу – графічний, операційна система – Linux Ubuntu.

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз ринку автоматизованних систем управління виробництвом
4.3 Вибір технологій для розробки програмного модуля
4.4 Розробка модулів
4.5 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 (*.ppt). – с. Формату А4.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт–ка
1	Аналіз сучасних систем та модулів керування та адміністрування підприємством ERP	05.05.2025	Виконано
2	Вибір технології для розробки модуля ERP	08.05.2025	Виконано
3	Проектування архітектури баз даних модуля	11.05.2025	Виконано
4	Розробка модуля адміністрування підприємством ERP	21.05.2025	Виконано
5	Оформлення пояснювальної записки	01.06.2025	Виконано
6	Подання роботи на рецензію	24.06.2025	Виконано

Дата видачі завдання 03 травня 2025 р.

Здобувач _____ Дмитро ЗАЛУЦЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

Керівник роботи _____ ст. викладач Дмитро ГУРІН
(підпис) (власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 75 с., 1 табл., 10 рис., 1 дод., 15 джерел.

ERP, АВТОМАТИЗАЦІЯ, СОБІВАРТІСТЬ, МОДУЛЬ УПРАВЛІННЯ, ЗАМОВЛЕННЯ, ПІДПРИЄМСТВО, КОМПОНЕНТИ, ОБЛАДНАННЯ, АРХІТЕКТУРА СЕРВІСІВ, LARAVEL.

Мета роботи – підвищення ефективності адміністрування виробничим підприємством шляхом створення автоматизованого програмного модуля з інтеграцією в наявну ERP–систему на основі сервісно–орієнтованої архітектури.

Об’єкт розробки – процеси внутрішнього адміністрування та управління замовленнями на виробничому підприємстві.

Предмет розробки – алгоритмічне, програмне та інформаційне забезпечення автоматичного модуля адміністрування підприємством.

У роботі описано повний цикл розробки програмного модуля для управління замовленнями, працівниками, обладнанням, матеріалами та розрахунком собівартості у виробничому середовищі. Проведено аналіз типових бізнес–процесів, існуючих систем управління та визначено архітектурні вимоги до модулю.

Розробку виконано з урахуванням Цілей сталого розвитку ООН, зокрема ЦСР 8 «Гідна праця та економічне зростання», ЦСР 9 «Індустріалізація, інновації та інфраструктура» та ЦСР 12 «Відповідальне споживання і виробництво».

У підсумку отримано стабільний та масштабований модуль, який дозволяє знизити витрати ручного адміністрування і покращити точність даних у системі підприємства.

ABSTRACT

Explanatory note: 75 pages, 1 tables, 10 figures,1 appendices, 25 sources.

ERP, AUTOMATION, COST, MANAGEMENT MODULE, ORDERS, ENTERPRISE, COMPONENTS, EQUIPMENT, SERVICE ARCHITECTURE, LARAVEL.

The goal of the project is to develop an automated enterprise administration module integrated into an existing ERP system using a service–oriented architecture.

Object of design – increase the efficiency of manufacturing enterprise administration by creating an automated software module with integration into an existing ERP system based on a service–oriented architecture. Subject of design – algorithmic, software, and informational solutions for an automatic enterprise management module.

The work describes the full cycle of developing a software module for managing orders, employees, equipment, materials, and cost calculations in a production environment.

A detailed analysis of business processes and existing ERP systems was carried out. Based on that, architectural requirements for the module were formed.

The development was carried out taking into account the UN Sustainable Development Goals, in particular SDG 8 “Decent work and economic growth”, SDG 9 “Industrialization, innovation and infrastructure” and SDG 12 “Responsible consumption and production”.

As a result, a stable and scalable system was built to reduce manual overhead and improve data accuracy within the enterprise information system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface (інтерфейс прикладного програмування);

Bitrix24 – корпоративна CRM/ERP-платформа;

BPMN – Business Process Model and Notation (нотація моделювання бізнес-процесів);

CRUD-інтерфейс – панель взаємодії для створення, перегляду, редагування та видалення записів;

CSV Comma-Separated Values (формат табличних даних із роздільниками);

CRUD –Create, Read, Update, Delete (основні операції з даними в базах);

DB – Database (база даних);

DTO – Data Transfer Object (об’єкт передавання даних між шарами програми);

ERP – Enterprise Resource Planning (система планування ресурсів підприємства);

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту);

JSON – JavaScript Object Notation (формат обміну даними між сервісами);

Laravel – PHP–фреймворк для розробки веб–додатків;

MVC – Model–View–Controller (архітектурна модель додатку);

MySQL – система керування реляційними базами даних;

ORM – Object–Relational Mapping (об’єктно–реляційне відображення);

REST – Representational State Transfer (архітектурний стиль побудови API);

SQL – Structured Query Language (мова структурованих запитів);

SQL-запит – інструкція для вибірки або обробки даних у базі;

UI – User Interface (користувацький інтерфейс);

UML – Unified Modeling Language (уніфікована мова моделювання);

UX – User Experience (досвід користувача).

ВСТУП

Сучасні реалії функціонування бізнес–структур характеризуються динамічним зростанням інформаційних потоків та ускладненням організаційно-управлінських процедур. У цих умовах традиційні методи керування підприємницькими структурами демонструють недостатню ефективність, що зумовлює необхідність пошуку інноваційних підходів до організації адміністративної діяльності.

Особливої актуальності набуває питання цифрової трансформації управлінських процесів, яка передбачає комплексне впровадження програмно–технічних засобів для автоматизації рутинних операцій та оптимізації прийняття стратегічних рішень. Відсутність ефективних інструментів автоматизованого управління призводить до зниження продуктивності праці адміністративного персоналу, збільшення ймовірності виникнення помилок у процесі обробки даних та неоптимального використання наявних ресурсів.

Проблематика розробки спеціалізованих програмних комплексів для адміністрування господарської діяльності підприємств охоплює широкий спектр технічних та методологічних питань. Зокрема, виникає необхідність створення інтегрованих рішень, здатних забезпечити централізоване управління різнорідними бізнес–процесами, ефективну координацію діяльності структурних підрозділів та формування аналітичної звітності в режимі реального часу.

Аналіз сучасного стану ринку програмних продуктів для автоматизації управління демонструє наявність певних прогалин у функціональних можливостях існуючих рішень. Більшість комерційних систем характеризуються високою вартістю впровадження, складністю налаштування під специфічні потреби конкретного підприємства та обмеженими можливостями інтеграції з існуючою IT-інфраструктурою.

У зв'язку з цим особливої важливості набуває завдання створення гнучкого та економічно доцільного програмного інструментарію, орієнтованого на комплексну автоматизацію адміністративних функцій підприємства. Такий інструментарій повинен забезпечувати можливість адаптації до специфічних особливостей організаційної структури та бізнес-процесів конкретного суб'єкта господарювання.

Впровадження автоматизованих систем управління сприяє досягненню цілого ряду стратегічних переваг: підвищенню швидкості обробки управлінської інформації, зменшенню трудовитрат на виконання адміністративних операцій, покращенню якості планування та контролю виконання бізнес–завдань, а також забезпеченню прозорості та підзвітності управлінських процесів.

Таким чином метою даною кваліфікаційної роботи є підвищення ефективності адміністрування виробничим підприємством шляхом створення автоматизованого програмного модуля з інтеграцією в наявну ERP–систему на основі сервісно-орієнтованої архітектури.

Об’єкт розробки – процеси внутрішнього адміністрування та управління замовленнями на виробничому підприємстві.

Предмет розробки – алгоритмічне, програмне та інформаційне забезпечення автоматичного модуля адміністрування підприємством.

– проаналізувати існуючі системи управління та автоматизації виробництвами;

– розробити структурну схему бази дпнних модуля;

– вибрати технології для розробки;

– розробити UI для панелі адміністрування;

– розроботит програмні складові.

Кваліфікаційну роботу виконано згідно з ДСТУ 3008–2015 [1] та керуючись навчальним посібником з дипломного проєктування [2] і відповідними методичними вказівками [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд ринку автоматизованих систем управління

У сучасну епоху цифрових трансформацій, коли інформаційні технології стрімко змінюють майже всі сфери діяльності людини, особливої актуальності набуває питання ефективного управління підприємствами. Це вже не розкіш, а необхідність, зумовлена як глобальними економічними процесами, так і внутрішніми потребами кожного окремого суб'єкта господарювання. Від дрібного виробника до транснаціональної корпорації – всі прагнуть до однієї мети: забезпечити прозоре, адаптивне й результативне управління ресурсами.

Автоматизовані системи управління підприємствами (АСУП) – це складні багаторівневі програмні комплекси, які мають на меті інтегрувати всі ключові процеси підприємства в єдине середовище. Це стосується фінансів, виробництва, кадрів, складу, постачання, збуту, логістики, документообігу, CRM, аналітики тощо.

Протягом останніх двадцяти років ринок таких систем зазнав величезної еволюції. Якщо раніше це були окремі програмки з обмеженим функціоналом, то сьогодні ми маємо справу з гігантськими ERP–платформами, що працюють у хмарі, підтримують мобільні додатки, інтегруються з BI–аналітикою та навіть використовують алгоритми штучного інтелекту.

Разом з тим, в Україні досі зберігається суттєвий розрив між потребами підприємств і можливостями наявних рішень. Чимало компаній продовжують працювати з морально застарілими локальними версіями систем, уникаючи оновлень через високу вартість або ризики збою. Інші, навпаки, впроваджують дорогі імпортні рішення, які в результаті не використовуються на повну потужність, бо не адаптовані до локального ринку, законодавства чи

менталітету персоналу.

1.2 Порівняння систем управління: теорія і практика

Система IT-Enterprise є одним із найпотужніших вітчизняних рішень для автоматизації управління підприємством у середовищі великих і середніх компаній. Вона вважається еталоном для побудови комплексної ERP-системи українського виробництва, особливо в галузях виробництва, логістики, управління проектами та фінансового обліку (рис. 1.2). Її основною перевагою є висока ступінь адаптивності, модульна структура, підтримка сучасних BPM- і MES-рішень, а також повна відповідність українському законодавству.

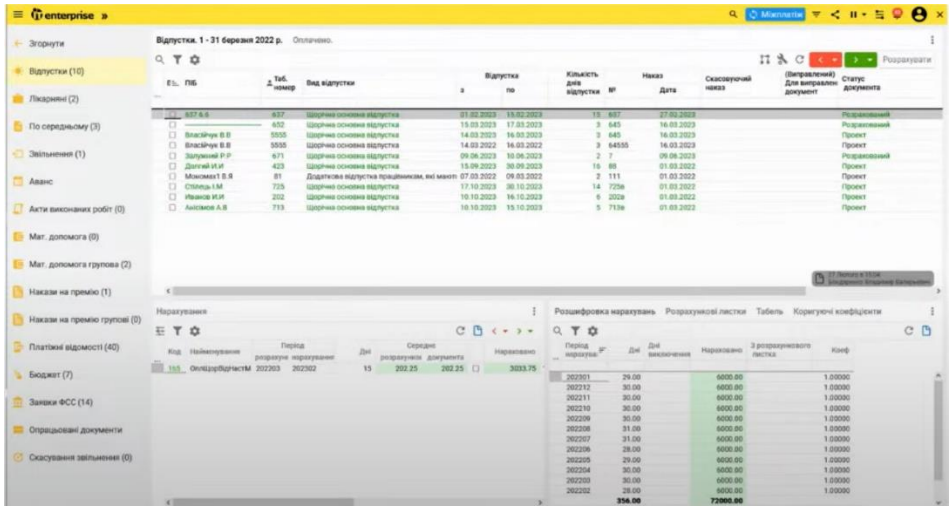

Рисунок 1.1 – Інтерфейс «1С: Підприємство»

Система BJET є сучасним українським рішенням для автоматизації управління підприємством, яке орієнтоване на середні та великі компанії, зокрема у сферах обліку, фінансів, логістики та управління ресурсами. Вона позиціонується як альтернатива іноземним ERP-системам, із гнучкою адаптацією до українського бізнес-середовища та нормативної бази (рис. 1.3). BJET побудована на базі платформи Odoo, що забезпечує відкритість, розширюваність і швидке впровадження необхідного функціоналу.

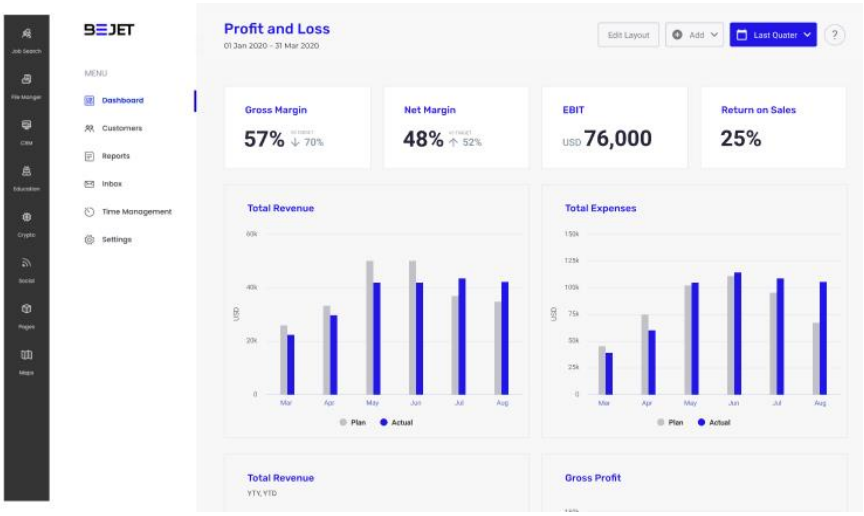

Рисунок 1.2 – Інтерфейс BJET

Система OneBox є універсальним українським рішенням для автоматизації бізнес-процесів, що поєднує функціональність CRM, ERP та BPM-систем. Вона особливо популярна серед малих і середніх підприємств, які прагнуть організувати роботу з клієнтами, управління завданнями, складським обліком, продажами та документообігом в єдиній платформі (рис. 1.4). Однією з ключових переваг OneBox є гнучка система налаштувань процесів без необхідності програмування — бізнес-логіка може бути адаптована безпосередньо через інтерфейс користувача.

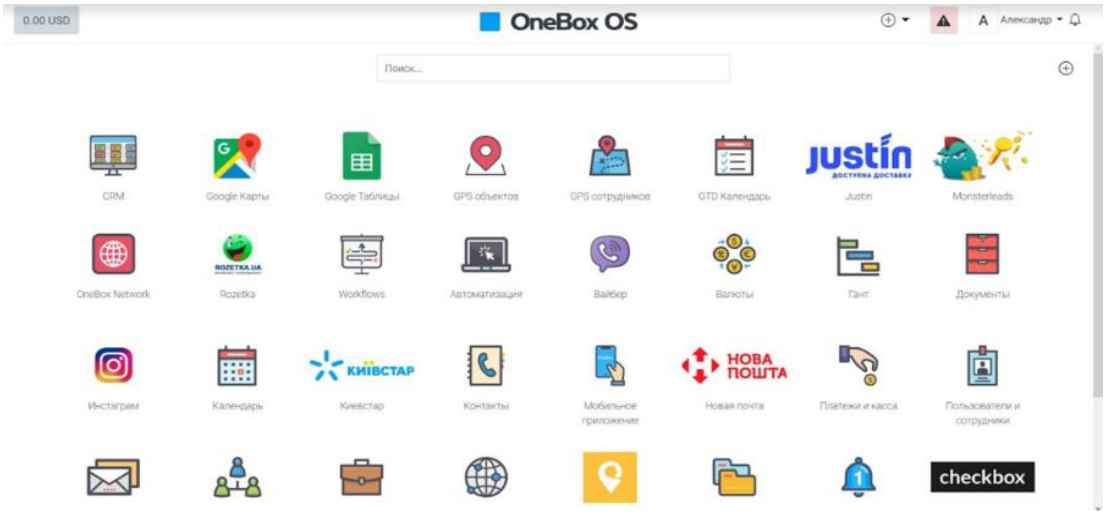

Рисунок 1.3 – Інтерфейс «OneBox»

На протилежному боці – важковаговики, такі як SAP [8], Oracle або Microsoft Dynamics. Ці системи – вершина інженерної думки, вони мають практично необмежений потенціал. Але разом із цим – величезні витрати на ліцензії, тривалі строки впровадження, залежність від великої кількості консультантів і фахівців, потреба у дорогій технічній інфраструктурі. Для багатьох українських підприємств такий підхід просто не є реалістичним у коротко – і середньостроковій перспективі.

1.3 Проблеми сучасних систем

На перший погляд може здатися, що автоматизована система – це панацея, яка вирішить усі проблеми підприємства. Але практика свідчить про протилежне. Дуже часто саме автоматизовані рішення стають джерелом нових труднощів:

– після впровадження системи кількість помилок у звітності зростає, оскільки персонал не встигає адаптуватися;

– автоматизовані звіти не відображають реальної картини, бо користувачі вводять дані формально;

– стандартизовані процедури вбивають гнучкість, яку раніше забезпечували досвідчені управлінці;

– інтеграція з іншими програмами виявляється складною або неможливою.

Причини таких невдач можна поділити на технічні, організаційні та психологічні. До технічних належать обмеження платформи, відсутність масштабованості, складність у налаштуваннях, неможливість адаптації під специфіку діяльності. Організаційні – це погане планування впровадження, відсутність єдиного бачення серед керівництва, конфлікт між «старою школою» та новими підходами. Психологічні ж пов’язані з тим, що люди бояться змін, не хочуть втрачати зону комфорту, а іноді й опираються

нововведенням саботажем.

Окремо варто згадати проблему надлишкової функціональності. Дуже часто система містить сотні модулів, з яких реально використовується лише 10–15%. Решта лише створює плутанину в інтерфейсі, займає ресурси та ускладнює навчання персоналу. Таке перевантаження, яке прийшло до нас з ідеології «all–in–one», не виправдане в умовах сучасного lean–менеджменту.

1.4 Висновок аналізу

Отже, можна констатувати, що ринок АСУП нині перебуває у стані глибокого парадоксу. З одного боку, існує безліч готових рішень. З іншого – підприємства не можуть знайти серед них те, яке б відповідало їхнім реальним потребам.

Це пояснюється низкою глибинних причин:

– переважна більшість рішень розроблялася для інших економічних реалій, ментальностей, правових систем;

– виробники систем не встигають за змінами у виробництві, ринку праці, цифрових платформа.

Саме тому на ринку з’являється запит на нове покоління програмного забезпечення – не просто ERP–систему, а інтелектуальний модуль, який:

– швидко адаптується до змін у законодавстві;

– працює з різними платформами (мобільні пристрої, веб, хмара);

– має дружній інтерфейс;

– не вимагає постійного обслуговування з боку програмістів;

– підтримує навчання користувача під час роботи (інтерактивні підказки, гіди);

– інтегрується з сучасними сервісами аналітики, документообігу, електронної пошти, месенджерів.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ

2.1 Вибір мови програмування та фреймворку: PHP та Laravel

При розробці сучасного веб-додатку для автоматизованого адміністрування підприємством було прийнято рішення використовувати стек PHP з фреймворком Laravel (рис. 2.1). Laravel є одним із найпопулярніших PHP–фреймворків, що поєднує високу швидкість розробки з гнучкістю та потужністю сучасної архітектури (рис. 2.2). Основні причини вибору:

– швидкий старт: завдяки команді Artisan розробник може генерувати шаблони моделей, контролерів та міграцій за лічені секунди;

– масштабованість: Laravel [11] дозволяє розробляти як невеликі модулі, так і складні багатокомпонентні системи (таблиця 2.1);

– безпека: вбудовані механізми захисту від SQL-ін’єкцій, XSS та CSRF;

– підтримка REST API: завдяки ресурсним контролерам реалізація API займає мінімум часу;

– велика спільнота: тисячі готових пакетів, документації, підтримка українською та англійською мовами;

– система черг та завдань: вбудований механізм Queue дозволяє виконувати ресурсомісткі операції в фоновому режимі, що значно покращує продуктивність додатку та користувацький досвід;

– кешування: підтримка різних драйверів кешування (Redis, Memcached, файлова система) для оптимізації швидкості роботи додатку та зменшення навантаження на базу даних;

– система подій та слухачів: гнучка архітектура для реалізації слабозв'язаного коду, що спрощує підтримку та розширення функціональності системи.

Таблиця 2.1 – Переваги обраного технічного стеку

Компонент	Перевага
Laravel	Швидкий старт, висока продуктивність, багата екосистема
PHP	Велика база розробників, хостингова доступність
MySQL	Надійність, простота адміністрування, масштабованість
DDD	Гнучка архітектура, адаптація до змін бізнесу, підтримка модульності
REST API	Легка інтеграція з мобільними застосунками, фронтендом, сторонніми сервісами

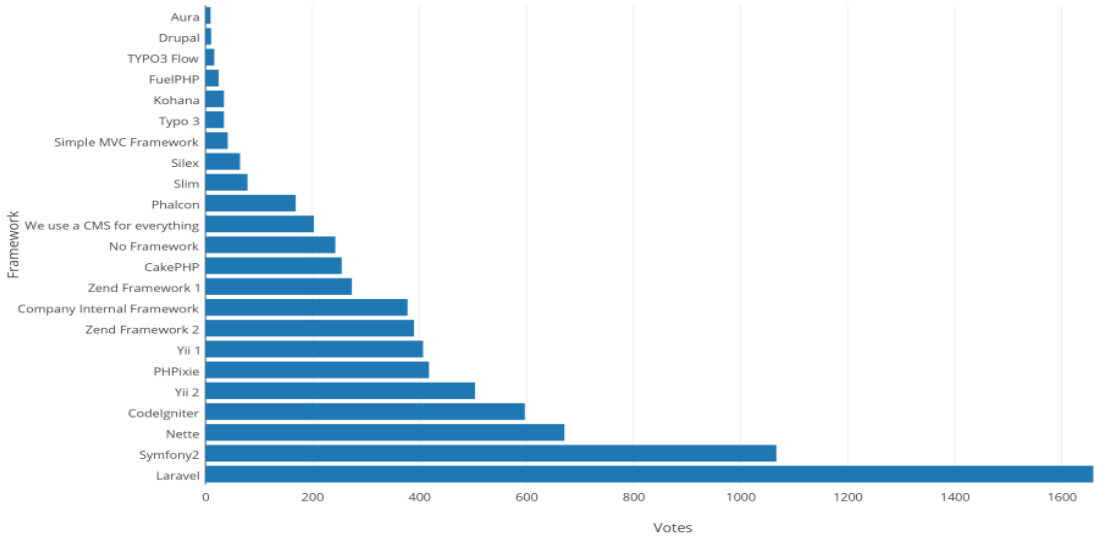

Рисунок 2.1 – Діаграма популярності PHP фреймоврков

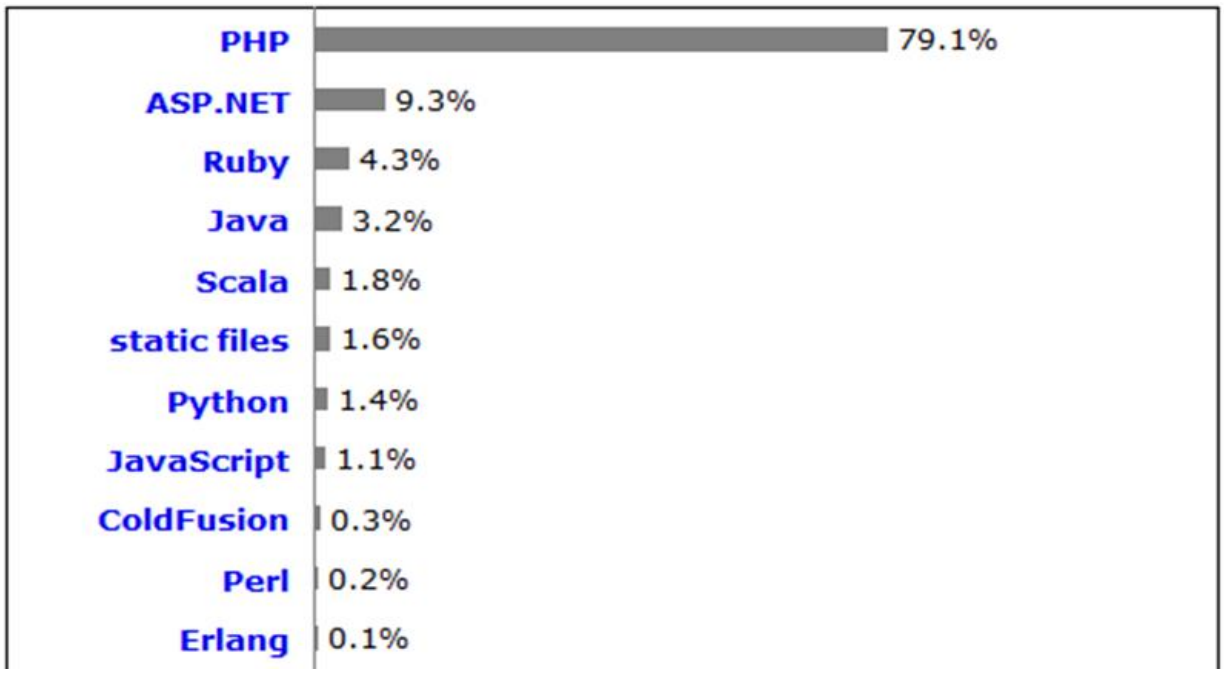

Рисунок 2.2 – популярності PHP у web–сайтах

2.2 Вибір бази даних: MySQL

MySQL – це реляційна система керування базами даних, яка широко використовується у комерційних і корпоративних проєктах [10]. Вона підтримується Laravel «з коробки», має чудову продуктивність при роботі з середніми й великими обсягами даних. Переваги MySQL у нашому контексті:

– висока продуктивність для аналітичних запитів і агрегованих таблиць;

– простота налаштування реплікації та масштабування;

– візуальні інструменти (phpMyAdmin, DBeaver) для адміністрування.

2.3 Організація архітектури: Domain–Driven Design [12] (DDD)

Однією з найважливіших частин нашого підходу стала реалізація програмної архітектури на основі принципів DDD – Domain-Driven Design [12]. Цей підхід дозволяє не просто писати код, а моделювати бізнес–

реальність у вигляді логічно зв’язаних сутностей.

Основні переваги DDD:

– чітке розділення відповідальностей: домен, застосування, інфраструктура, інтерфейс – кожен шар виконує свою роль;

– зрозумілість коду: кожна модель відображає реальний об’єкт ("Замовлення", "Користувач", "Заявка"), що зменшує плутанину в логіці;

– гнучкість у розширенні: легко додавати нові бізнес-правила, не порушуючи існуючу структуру;

– тестованість: DDD дає змогу розробляти покриті юніт-тестами доменні сценарії, незалежно від інтерфейсу або бази даних (рис. 2.3).

Модель застосування виглядає наступним чином: Domain layer – описує бізнес–логіку та інтерфейси домену. Application layer – координує логіку, використовуючи сервіси. Infrastructure layer – реалізує доступ до бази, API, файлової системи. Interface layer – контролери, HTTP–запити, UI.

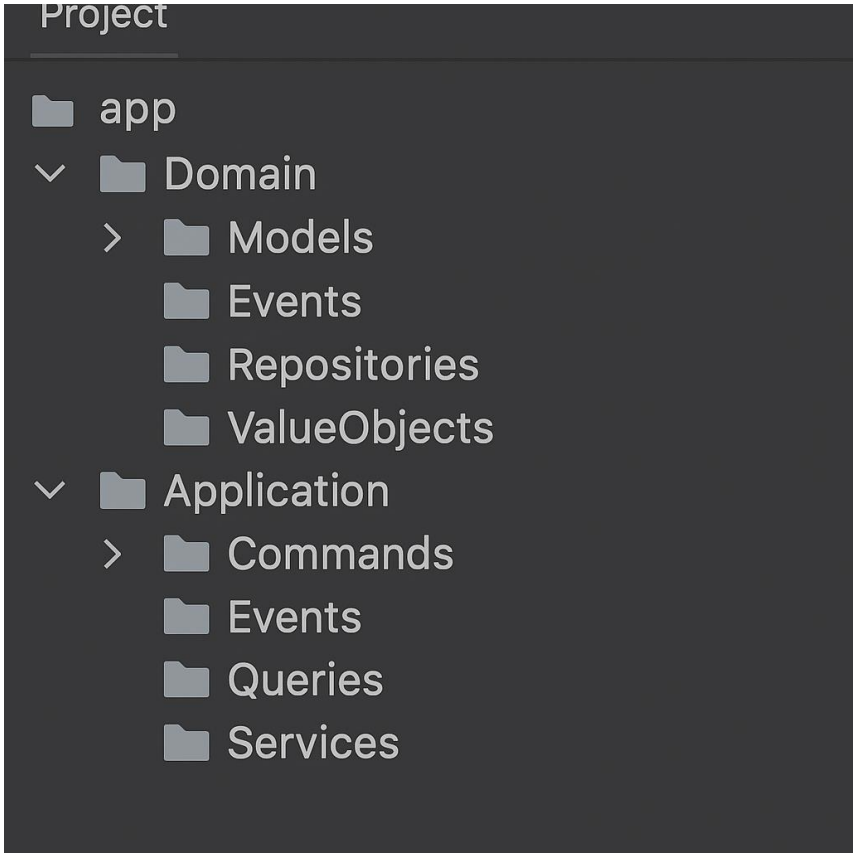

Рисунок 2.3 – Схема архітектури DDD у laravel

2.4 Альтернативні фреймворки

На поточний момент існують не менш популярні технології розробки:

– Node.js/Express – висока продуктивність, але складніше підтримувати у великих проектах;

– Python/Django – потужна ORM, але слабша підтримка DDD;

– PostgreSQL – потужніший за MySQL для складної аналітики, але потребує більше навичок для адміністрування;

– Monolithic architecture – швидкий старт, але важко масштабувати, розділяти команди.

Резюмуючи: вибраний підхід – це оптимальний баланс між швидкістю розробки, масштабованістю, простотою підтримки та відповідністю задачам підприємства.

2.5 Поглиблене обґрунтування вибору Laravel

Laravel підтримує шаблон MVC (Model-View-Controller), що дозволяє розділити логіку додатку, представлення й керування даними. Завдяки цьому розробка стає більш організованою, що критично при командній роботі над великими проектами.

Окрім цього, Laravel має зручний механізм роботи з міграціями бази даних. Це дозволяє ефективно керувати структурою бази без необхідності ручного створення SQL–запитів. Laravel Eloquent ORM забезпечує зрозумілу модель доступу до даних, завдяки чому нові розробники можуть швидко орієнтуватися у проекті.

Laravel також пропонує такі можливості, як:

– Queue system – для обробки фонових завдань (відправка листів, синхронізація з API);

– Broadcasting – реальний час, нотифікації, WebSockets;

– Scheduler – вбудований планувальник задач (альтернатива cron);

– Sanctum/Passport – інструменти аутентифікації та авторизації для побудови безпечних REST API.

2.6 Архітектурна схема програмного рішення

Нижче наведено загальну архітектурну схему системи, яка базується на DDD і Laravel [11].

Компоненти:

– Frontend (React/Vue) – взаємодіє з REST API;

– API Gateway – контролює авторизацію, маршрутизацію, вхідні запити;

– Application Layer – оркеструє сервіси та бізнес–логіку;

– Domain Layer – містить сутності, агрегати, value-об’єкти, інтерфейси репозиторіїв;

– Infrastructure – MySQL, файлові сховища, сторонні API (наприклад, Telegram Bot API) [10].

2.7 Взаємодія з мобільним додатком

Завдяки REST–архітектурі, система дозволяє реалізувати повноцінну клієнтську частину на Android/iOS. Кожен запит з мобільного додатку виконується через авторизовану сесію, використовуючи JWT або Laravel [11] Sanctum. Це забезпечує як безпеку, так і масштабованість.

Наприклад, робочий може отримати свої задачі через GET /api/tasks, оновити статус через POST /api/tasks/{id}/update, або отримати push–нотифікацію через Firebase.

Проєкт орієнтований на DevOps–підхід. Використовується GitLab CI/CD або GitHub Actions для автоматичного розгортання:

– при пуші в main відбувається збірка контейнерів docker;

– laravel запускає міграції бази;

– нотифікація надходить до telegram-каналу команди.

Система спроєктована таким чином, щоб у майбутньому можна було:

– підключити модулі bi–аналітики (metabase, superset);

– інтегрувати чат–бота у telegram, whatsapp;

– зберігати документи в хмарі (google drive, s3);

– масштабувати базу за рахунок реплікації mysql або переходу на postgresql;

– перейти до мікросервісної архітектури без змін у доменних моделях.

У результаті отримуємо гнучку, безпечну, масштабовану систему, яка має потенціал перетворитися на повноцінну ERP–платформу для українських і міжнародних підприємств.

3 ФУНКЦІОНАЛЬНІ МОДУЛІ СИСТЕМИ ТА СЦЕНАРІЇ ВИКОРИСТАННЯ

3.1 Загальна структура модулів

У розробленій системі адміністрування передбачено реалізацію кількох функціональних модулів, кожен з яких відповідає за окремий сегмент управління підприємством. Модулі реалізовані згідно з принципами розділення відповідальностей, із чіткою межою між бізнес-логікою, API, базою даних та інтерфейсом (рис. 3.1).

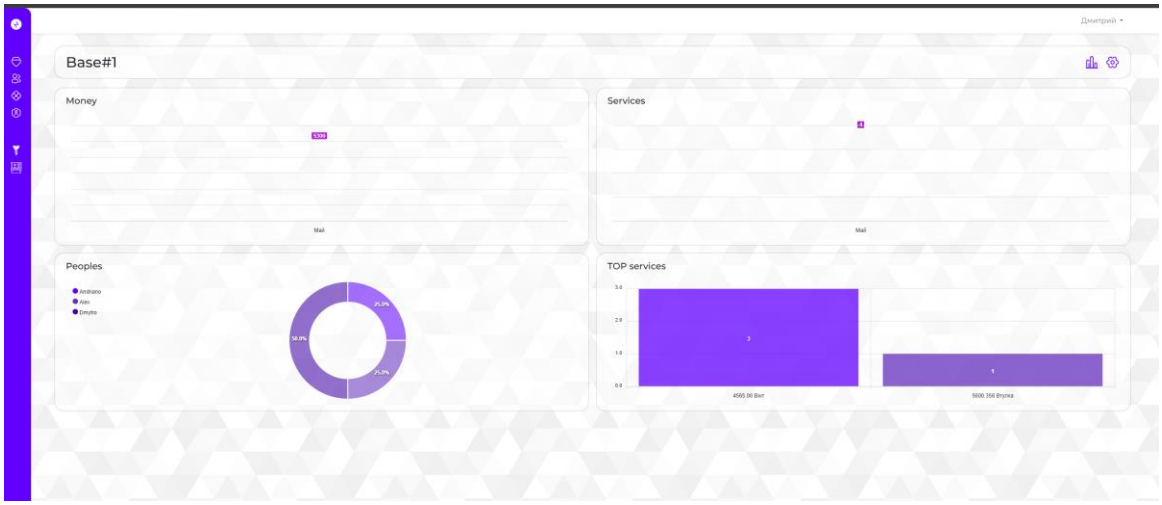

Рисунок 3.1 – Загальний вид інтрефейсу

3.2 Модуль кадрового обліку

Цей модуль відповідає за реєстрацію працівників, зберігання персональних даних, призначення ролей і прав доступу. Передбачено також інтеграцію з системою контролю доступу (СКУД), що дозволяє відстежувати присутність працівників на об’єкті.

Основні функції:

– реєстрація нового працівника;
– призначення посад, відділів, підрозділів;
– ведення журналу руху кадрів (прийом, звільнення, переведення);
– формування штатного розпису.

3.3 Модуль заявок і маршрутів узгодження

Модуль забезпечує створення, редагування, погодження й контроль за виконанням заявок різного типу (на закупівлю, ремонт, аудит, перевірку якості тощо). Передбачено гнучке налаштування маршрутів узгодження за ролями.

Сценарій використання:

– користувач створює заявку, додає фото, коментар;
– система автоматично формує маршрут узгодження (наприклад: начальник цеху → головний інженер → фінансовий директор);
– учасники отримують сповіщення та затверджують або повертають заявку;
– при завершенні – генерується виконавче доручення.

3.4 Модуль контролю операцій

Центральна частина системи для виробничих підприємств. Відповідає за реєстрацію та контроль проходження технологічних операцій, призначення виконавців, контроль відхилень і браку.

Функції:

– призначення операцій за маршрутом;
– завантаження технічних інструкцій;
– облік браку, доопрацювань, простоїв;
– інтеграція з терміналами/смартфонами для реєстрації QR–кодів.

3.5 Журнали та аналітика

Система формує звіти на основі журналів подій, а також візуалізує аналітичні дані у вигляді графіків, зведених таблиць, KPI та індикаторів. Аналітичний інструментарій побудовано на агрегованих запитах до MySQL із можливістю експортів у Excel, PDF, Google Таблиці.

Аналітика включає:

– ефективність кожного працівника;
– відхилення по виробництву за зміну/добу/тиждень;
– навантаження на відділи;
– прогнозне навантаження (на основі історичних даних).

4 РОЗРОБКА МОДУЛІВ

4.1 Модуль «Співробітники»: створення домену та обліку кадрів

У процесі побудови цифрової системи управління підприємством, першим логічним кроком стало створення повноцінного кадрового модуля. Саме люди, а не процеси чи обладнання, є центральною ланкою будь-якої виробничої системи. Кожне завдання, кожна операція на верстаті, кожна заявка і погодження – усе виконується, створюється або контролюється людиною. Тому на старті розробки важливо було закласти стійкий, масштабований фундамент обліку персоналу. Структура домену була побудована так, щоб відображати як організаційні особливості, так і логіку взаємодії між ролями, підрозділами, обов’язками та рівнем доступу. Кожна людина в системі представлена не лише як запис у базі даних, а як активна сутність, яка бере участь у виробничих і адміністративних процесах.

Основна бізнес-ідея полягала у створенні цифрового профілю працівника, який би замінював паперову особову справу. Такий профіль мав містити ключові дані: повне ім’я, службовий код, посаду, телефон, email, дату прийому, статус активності, а також – список історичних подій: переведення, участь у замовленнях, пропуски, відпустки, заміни. Всі ці параметри в майбутньому стали джерелом для аналітичних звітів, а також – основою для нарахування трудової частини собівартості. Іншими словами, модуль «Співробітники» не обмежувався формальною карткою кадрів, а був повноцінним аналітичним, інтеграційним та контрольним інструментом.

Розробка почалася з побудови таблиць у базі даних: основна таблиця працівників, таблиця посад, історія змін, таблиця пов’язування з обліковими записами. Кожен працівник отримував унікальний ідентифікатор, який використовувався в інших модулях як зовнішній ключ: наприклад, в

замовленнях, завданнях або логах операцій. Далі розпочався етап створення доменних моделей – це були класи, які описували правила поведінки об’єкта «співробітник»: заборона видалення активного користувача, логіка автоматичного прив’язування до цеху за посадою, правила унікальності службових кодів тощо.

На рівні API були реалізовані ендпоінти для отримання списку працівників, створення, редагування, відображення історії дій. Важливо, що кожна зміна – наприклад, зміна посади чи деактивація – записувалася у лог змін. Це дозволяло бачити не тільки поточний стан, але й відслідковувати динаміку кадрових змін. На фронтенді був реалізований інтерфейс із табличним переглядом, можливістю фільтрувати по ролях, підрозділах, статусу, а також – відображенням аналітики за участю працівника в замовленнях, його навантаженням, середнім часом виконання завдань.

У рамках цього модуля окрему увагу було приділено безпеці даних. Доступ до перегляду і редагування мав лише користувач з відповідними правами. Була реалізована повноцінна система ролей: HR-менеджер міг редагувати співробітників усіх цехів, але цеховий майстер – тільки працівників свого підрозділу. Усі запити до API проходили через middleware, які перевіряли не лише JWT–токен, а й наявність дозволу до конкретної групи даних.

Після реалізації базових функцій система була інтегрована з Telegram-ботом. Кожному працівнику можна було надсилати повідомлення – наприклад, нагадування про чергову зміну, прохання підтвердити присутність або сповіщення про нове завдання. Також, через цей бот працівник міг повідомити про відсутність, що миттєво відображалося в інтерфейсі диспетчера і автоматично знімало його з запланованого завдання.

У фінальному вигляді модуль «Співробітники» став не лише технічно завершеним, а й повністю інтегрованим із рештою системи. Його дані безпосередньо впливають на собівартість, маршрути погодження, розподіл

ролей і доступів, а також – дозволяють створити основу для майбутньої системи мотивації, преміювання та ефективності праці. Саме з цього модуля починається будь-який облік, планування та контроль.

Першим модулем, який почав розроблятися, був модуль управління працівниками (рис. 4.1). Цей вибір був обумовлений тим, що саме працівники є базовою одиницею в усіх процесах: вони виконують замовлення, працюють за станками, формують трудові витрати. Модуль розпочався з проєктування сутностей у домені HR. Були визначені поля, які повинна містити кожна картка працівника: повне ім’я, код, електронна пошта, телефон, посада, дата прийому, статус активності. Також були створені таблиці для обліку ролей, історії змін посад, участі у виробничих процесах. Після цього розроблявся інтерфейс створення, перегляду і редагування працівників. Важливо, що з перших днів передбачалась логіка пов’язування працівника з користувачем системи через єдиний обліковий запис. Уся перевірка прав доступу базується на цьому зв’язку.

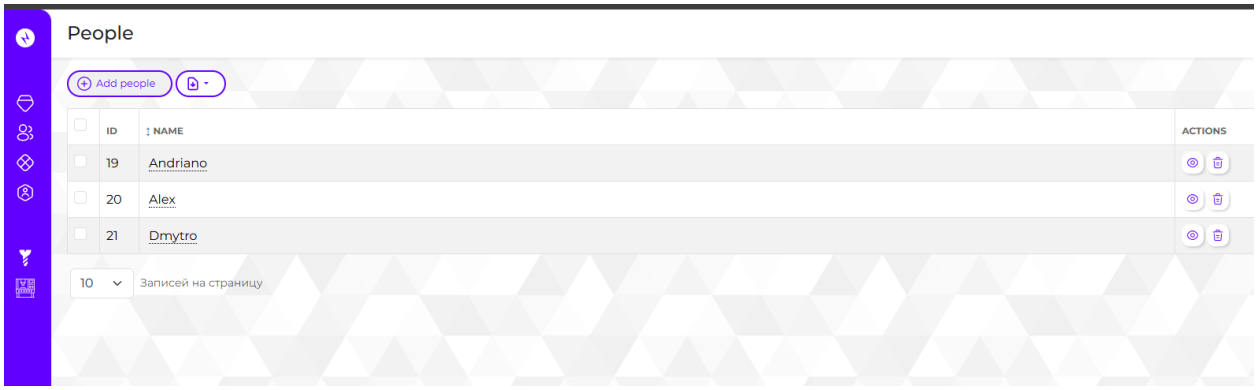

Рисунок 4.1 – Розділ «Співробітники»

4.2 Розділ «Обладнання»: цифровий паспорт виробничих потужностей

Модуль управління обладнанням став другим за пріоритетністю після кадрової частини системи. Його роль полягає в тому, щоб забезпечити точний, глибокий та структурований облік усього верстатного фонду підприємства. У

сучасному виробництві обладнання є не лише засобом виготовлення продукції, а й активом, що формує витрати. Кожна хвилина простою – це потенційна втрата, а кожна хвилина перевантаження – ризик поломки. Саме тому від точності, своєчасності та достовірності даних про обладнання залежить як ритмічність виробничого процесу, так і фінансова стабільність підприємства.

На початковому етапі розробки було проведено аудит верстатного парку: сформовано перелік одиниць обладнання, їх типи, характеристики, розташування, обслуговуючий персонал. Всі ці дані стали базою для створення структури домену "Обладнання". Були визначені основні атрибути: унікальний код верстата, модель, виробник, рік виготовлення, дата введення в експлуатацію, технічний стан, поточне місце встановлення, цех або зона. Також була реалізована класифікація за типом (токарний, фрезерний, оброблювальний центр, універсальний, ручний, автоматизований), що дозволило надалі формувати аналітику по групах.

У базі даних була створена таблиця machines, до якої додавалися пов’язані таблиці: технічні характеристики, історія переміщень, історія обслуговування, лог використання, історія аварій і збоїв. При створенні моделі "Обладнання" в Laravel [11] були реалізовані методи для автоматичного підрахунку поточного навантаження, визначення коефіцієнта простою, розрахунку амортизаційної вартості.

Особливу увагу було приділено реалізації логіки журналу роботи верстата. Кожного разу, коли оператор запускає або завершує завдання, система фіксує подію в machine_logs. Цей лог зберігає дату і час, тривалість, ідентифікатор замовлення, ПІБ оператора, тип операції (внутрішня обробка, пробна партія, доопрацювання, серійне виробництво). Саме на основі цих записів обчислюється машинна трудомісткість, яка потім використовується у фінансовій частині калькуляції.

Інтерфейс модуля розроблявся в тісній співпраці з інженерами і

диспетчерами. Було враховано побажання відображати ключові показники відразу на екрані: статус верстата (в роботі, в ремонті, в очікуванні), кількість завдань у черзі, оператор, час останнього запуску. Також була створена інтерактивна візуалізація завантаження верстатів у вигляді графіка по змінах: користувач міг вибрати день або тиждень і побачити розклад роботи кожної одиниці обладнання, включаючи накладання задач, простої, неефективні вікна.

Окремим компонентом стала система аналізу ефективності. Було реалізовано алгоритм підрахунку коефіцієнта ефективного використання обладнання (OEE): співвідношення фактичного часу роботи до календарного, з урахуванням планових і аварійних простоїв. Цей індикатор дозволяв оперативно виявляти проблеми у виробництві. Наприклад, якщо верстат працює лише 3 години на зміну при запланованих 6, це сигналізує про недостатнє завантаження або проблеми з матеріалами чи персоналом. Усі ці дані виводилися в окремий дашборд.

Було також реалізовано функцію автоматичного формування попереджень про необхідність технічного обслуговування. Після кожних 100 або 500 мотогодин (залежно від типу обладнання) система створювала запис у журналі техобслуговування, інформувала відповідального інженера та блокувала нові виробничі завдання, поки верстат не пройде сервіс. Цей підхід дозволив підвищити надійність парку обладнання та зменшити кількість аварій.

На рівні API були створені маршрути для перегляду списку обладнання, оновлення статусу, створення нової одиниці, перегляду журналу, генерації звітів по ефективності. Доступ до цих функцій контролювався за допомогою middleware на основі ролей: технік міг оновлювати логи, інженер – переглядати звіти, а адміністратор – вносити зміни до конфігурації.

Завдяки високому ступеню деталізації, модуль «Обладнання» став не просто реєстром верстатів, а цифровим відображенням технічного ресурсу

підприємства. Його інтеграція з іншими модулями дала змогу будувати повну виробничу картину: від завантаження змін до впливу на собівартість. На базі цього модуля в подальшому планується реалізація прогнозування зносу та автоматичного планування ремонтів.

4.3 Розділ «Замовлення»: логіка виробничого процесу

Модуль замовлень у структурі системи управління підприємством займає одну з найважливіших позицій. Його значення полягає не лише у формальному створенні виробничих заявок, а й у глибокій інтеграції з усіма іншими бізнес–процесами: розрахунками, кадровими ресурсами, матеріалами, станками, аналітикою. Воно включає технічну частину (що потрібно виготовити), часову (дедлайн), людську (хто виконує), машинну (де буде виконано) та економічну (яка собівартість і вартість реалізації).

Розробка модуля почалась із формалізації сутності "Замовлення". У системі кожне замовлення має свій унікальний номер, дату створення, опис, відповідального ініціатора, а також перелік операцій, які потрібно виконати в межах цього замовлення (рис. 4.2). Одразу були визначені атрибути: назва замовлення, опис, пов’язаний клієнт або підрозділ, технічні вимоги, термін виконання, статус (у черзі, виконується, завершено, скасовано), пріоритет, маршрут узгодження, витрати (матеріали, праця, станки), а також вартість.

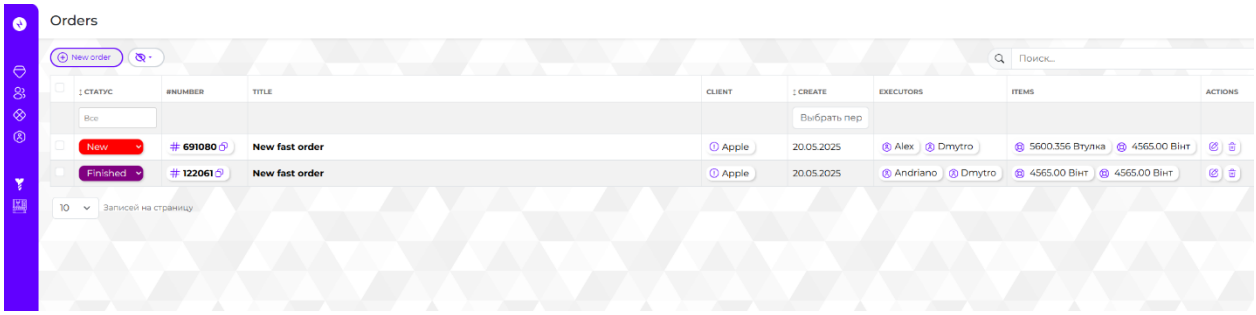

Рисунок 4.2 – Розділ Замовлення

На рівні бази даних було створено таблицю orders, а також допоміжні order_items, order_routes, order_tasks, order_logs. Така розгалужена структура дозволяє деталізувати як склад замовлення, так і його внутрішні етапи. Наприклад, одне замовлення може містити три деталі, кожна з яких має свої технологічні маршрути, залучає різні верстати і виконавців. У таблиці order_routes зберігається інформація про те, які ролі або конкретні особи повинні погодити замовлення перед запуском. Це важливо для дотримання виробничої дисципліни, контролю бюджету і відповідальності.

Інтерфейс замовлення розроблений з урахуванням потреб трьох основних категорій користувачів: виробничих менеджерів, майстрів цехів і керівництва. Для менеджера – це можливість швидко створити нове замовлення, вибрати необхідні деталі, додати технологічну документацію, вказати виконавців. Для майстра – це можливість бачити актуальні задачі, призначити змінного виконавця, контролювати виконання поетапно. Для керівника – це фінансове зведення і контроль статусів. Усі ці режими реалізовані через один екран, де в залежності від ролі система автоматично змінює доступні дії.

Важливим етапом стало впровадження системи маршрутів узгодження. Перед тим як замовлення переходить у статус «до виконання», воно проходить через погодження: спочатку майстер, потім технолог, іноді – бухгалтер або фінансовий директор. Це дозволяє перевірити технічну реалістичність замовлення, його відповідність бюджету, наявність ресурсів. Кожне погодження фіксується із зазначенням дати, статусу, коментаря. У випадку відхилення – замовлення повертається ініціатору із поясненням.

На API–частині були реалізовані ендпоінти: створення, перегляд, оновлення, скасування, погодження, формування фінансового звіту, підрахунок собівартості. Усі дії логуються – як із фронтенду, так і з мобільного додатку. Це дозволяє у будь–який момент відновити хронологію подій або підтвердити дію відповідальної особи.

У структурі замовлення особливу увагу приділено відображенню витрат (рис. 4.3). Система автоматично підтягує всі пов’язані витрати: кількість відпрацьованих хвилин на верстаті, тривалість людської участі, використані матеріали. На основі цих даних формується повна калькуляція. У фінансовій панелі кожного замовлення відображається собівартість (розбита по категоріях), ціна реалізації, потенційний прибуток, відхилення від нормативу. Це дозволяє керівництву бачити не лише технічне виконання, а й економічний результат кожного замовлення.

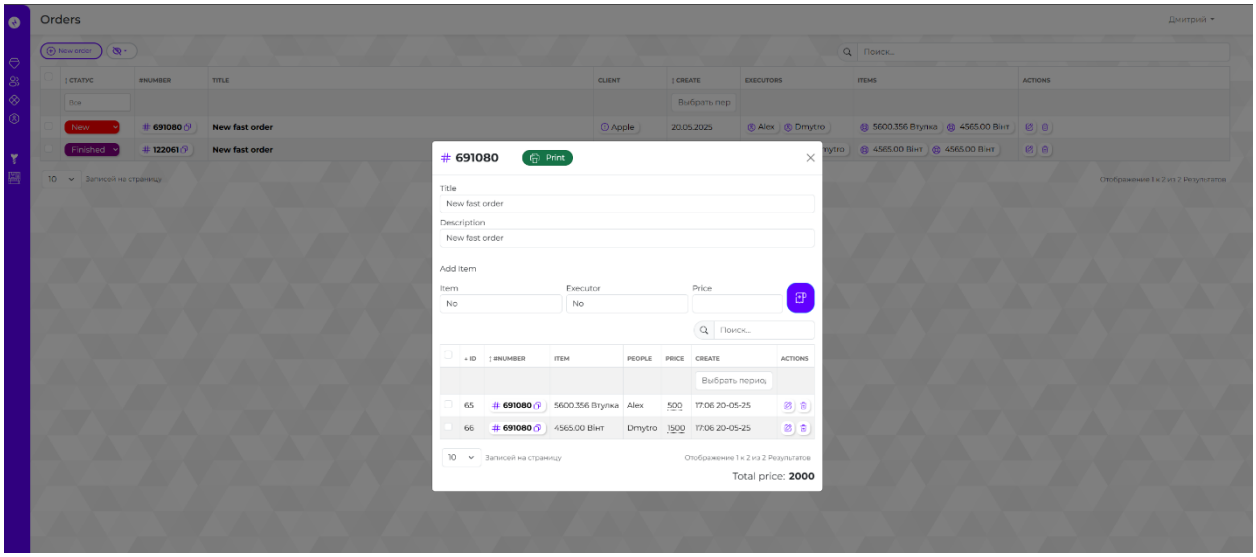

Рисунок 4.3 – Вікно редагування замовлення

Розширена аналітика дозволяє формувати звіти за замовленнями по періоду, виконавцях, типах операцій, відділах. Також реалізовано панель контролю прострочених замовлень, замовлень без погодження, замовлень із перевищенням бюджету. Усі ці звіти доступні в інтерфейсі і через експорт у форматах Excel, PDF, Google Sheets.

Таким чином, модуль «Замовлення» став ядром операційної частини системи, зв’язуючи людей, обладнання, матеріали й економіку. Саме через нього реалізується логіка запуску виробництва, оцінка ефективності, дисципліна виконання. Його універсальність дозволяє використовувати його

як у серійному виробництві, так і у разових замовленнях або внутрішніх сервісних задачах. Подальші плани розвитку модуля включають інтеграцію з календарем, автоматичне планування зміни, API для мобільних терміналів і можливість завантаження ТЗ у PDF або CAD–форматах.

Модуль замовлень був спроєктований як центральна ланка між усіма об'єктами системи. Кожне замовлення містить заголовок, опис, відповідального користувача, цільовий термін виконання, список операцій, пов’язаних працівників і технічні вимоги. Було реалізовано логіку маршруту проходження замовлення: від ініціації до завершення з обов’язковими погодженнями на кожному етапі. Також впроваджена можливість фіксувати завдання, які виконуються в рамках замовлення, і автоматично підраховувати витрачені ресурси – як по людських годинах, так і по хвилинах роботи обладнання.

4.4 Розділ «Матеріали»: логіка обліку, витрат і вартості

У системі цифрового управління виробничими процесами облік матеріалів виконує ключову роль – він визначає основу будь-якої калькуляції, формує зв’язок між закупівлею та виробництвом, контролює складські залишки та прогнозує майбутню потребу. Початок розробки модуля «Матеріали» ґрунтувався на ідеї повного переходу від неструктурованих списків, Excel-таблиць і ручних виписок до централізованого та логічно збудованого цифрового середовища. Основним завданням стало створення такої моделі даних, яка не лише зберігала би інформацію про найменування, одиницю виміру та ціну, а й дозволяла би відстежувати історію використання, цінові зміни, вплив на собівартість та зв’язок з кожною окремою операцією.

Початково було створено таблицю materials, яка містила унікальний ідентифікатор матеріалу, його назву, тип, одиницю виміру (метр, кілограм, лист, літр), поточну ціну, категорію, критичний залишок, статус (активний або

архівний). Окремо була створена таблиця material_movements, яка відповідала за всі події переміщення: надходження, списання, повернення, переміщення між складами. Для кожного рядка зберігались дата, тип дії, кількість, відповідальна особа та пов’язаний документ (замовлення, заявка, акт).

Центральним моментом у проєктуванні була не лише архітектура, а й логіка зв’язків. Кожен матеріал може бути прив’язаний до одного або кількох етапів виробництва, до конкретного замовлення, або навіть до специфікації певного типу продукції. Було реалізовано механізм так званих «виробничих наборів» – коли під час створення замовлення інженер вибирає тип деталі, система автоматично пропонує список необхідних матеріалів з розрахованими нормативами. Це забезпечує стандартизацію виробництва, а також дозволяє порівнювати планові та фактичні витрати.

Крім базового обліку, система містить модуль аналізу ціни. Усі зміни вартості матеріалів фіксуються у таблиці material_prices, де зазначено джерело зміни (нове надходження, коригування бухгалтерії, зміна ринку), дату та відповідального користувача. Це дозволяє будувати графіки динаміки, прогнозувати коливання вартості й фіксувати вплив на собівартість готової продукції. Такий підхід особливо важливий у випадках, коли закупівля матеріалів має імпортну складову або залежить від коливань валют.

Особливе місце у модулі займає логіка критичного залишку. Для кожного матеріалу можна задати мінімальний рівень запасу. У разі його досягнення система автоматично формує попередження у панелі диспетчера, а також надсилає повідомлення у Telegram–бот відповідального за закупівлі. Така автоматизація виключає людський фактор і мінімізує ризики зупинки виробництва через відсутність ресурсів.

Інтерфейс модуля містить декілька режимів: перегляд загального каталогу, перегляд історії руху конкретного матеріалу, режим актуалізації ціни, режим резервування. Окремо реалізовано функціональність підбору заміни – якщо для конкретного матеріалу існують дозволені альтернативи,

система пропонує їх у випадку дефіциту. Усі операції над матеріалами логуються – від створення до списання, з фіксацією IP, браузера, ролі користувача.

На рівні API модуль реалізований як окремий ресурс із маршрутизованими запитами REST. Реалізовані ендпоінти для створення матеріалу, редагування, перегляду, додавання до замовлення, формування звіту, вивантаження залишків у Excel. Інтеграція з мобільним додатком дозволяє працівнику складу сканувати QR–код матеріалу й моментально бачити всю інформацію: опис, залишок, резерви, переміщення. Також можливо здійснювати списання чи переміщення зі смартфона.

У підсумку модуль «Матеріали» став не лише каталогом, а повноцінним SCM–компонентом (Supply Chain Management). Він охоплює закупівлі, внутрішній облік, планування та списання. У перспективі він буде інтегрований з бухгалтерією, MRP–модулем (планування потреб) і зовнішніми ERP для імпорту рахунків.

4.5 Реалізація модуля собівартості в практиці

4.5.1 Передумови впровадження

На момент початку практичної реалізації модуля собівартості система вже містила ключові функціональні блоки, які забезпечували централізований облік персоналу, обладнання, матеріалів та замовлень. Це дозволило створити основу для побудови аналітичної системи, яка оперує не абстрактними значеннями, а конкретними, перевіреними в реальному часі даними. Факт наявності зв'язаної моделі даних відкрив можливість для агрегування витрат на кожному етапі виробничого процесу. Структура собівартості має розгалужену природу – це не просто арифметична сума, а результат інтегрованого аналізу діяльності підприємства. У її розрахунку враховуються не лише кількість і ціна матеріалів, а й машинний час, людські витрати,

накладні витрати, коефіцієнти коригування, норми, а також індекси ризику або інфляційного коливання.

Впровадження цього модуля передбачало зміну способу мислення керівників середньої ланки. Якщо раніше основним критерієм оцінки ефективності вважалась кількість вироблених одиниць або швидкість виконання, то тепер додалась нова категорія – фінансова доцільність. Це означає, що об'єктивно дорожче замовлення може бути менш вигідним у порівнянні з менш капіталоємким, але регулярним. Для цього система мала адаптуватися до реального стану підприємства. На початковому етапі були сформульовані вимоги: інтеграція даних без дублювання, єдина точка обчислення собівартості, можливість експорту результатів, розділення доступів, адаптивна логіка для різних типів замовлень.

Технічною передумовою стало використання сервіс–орієнтованої архітектури. Всі попередні модулі, реалізовані в межах доменів, були адаптовані до ролі постачальників даних. Наприклад, модуль «обладнання» експортував не лише список верстатів, а і журнал часу з тарифами, прив’язаними до замовлень. «Співробітники» передавали ставки працівників і тривалість участі. «Матеріали» – розрахункову ціну одиниці та витрати на кожну операцію. Завдяки цьому, модуль собівартості виконував роль агрегатора, не створюючи нових джерел даних, а використовуючи існуючі як базу для фінансових розрахунків.

Важливо, що під час формування технічного завдання передбачалась масштабованість – тобто можливість підключення додаткових коефіцієнтів витрат у майбутньому. Наприклад, електроенергія, водопостачання, оренда площі або податки. Усі ці параметри можуть бути підключені як окремі сервіси або таблиці з параметричним впливом. Перший прототип модуля було протестовано на тестовому середовищі з 12 реальними замовленнями, після чого були зібрані помилки, оцінена точність і фінансові відхилення між «ручним» та «системним» обрахунком. За результатами виявлено похибку в

межах 2,3%, що було визнано допустимим у межах галузевої похибки та стало підставою для запуску на повний обсяг виробництва.

На момент початку практичного етапу розробки система мала сформовані та інтегровані модулі: «співробітники», «обладнання», «матеріали» та «замовлення». Кожен із них генерував структуровані, уніфіковані дані. Це стало базовою умовою для реалізації модуля «собівартість», який опирається на логіку крос–модульної агрегації витрат.

4.5.2 Архітектурне впровадження розрахунку

Після визначення бізнес-логіки формування собівартості наступним етапом стало архітектурне впровадження обчислювального механізму в структуру всієї ERP–системи. Було важливо побудувати розрахунок так, щоб він не створював дублювання даних, не порушував принципу єдиної точки істини (single source of truth), а також щоб його можна було легко масштабувати, адаптувати і викликати як у ручному режимі, так і автоматично при зміні стану замовлення. Через це було ухвалено рішення реалізувати розрахунок собівартості у вигляді сервісу в рамках доменної архітектури з винесенням логіки до окремого класу CostCalculationService.

Цей сервіс став логічним центром, який агрегував дані з трьох основних джерел: репозиторію працівників, репозиторію обладнання та репозиторію матеріалів. Всі вони були попередньо реалізовані у вигляді окремих класів з методами доступу до пов’язаних із замовленням записів. Наприклад, для певного замовлення сервіс отримував перелік завдань і через orderTasksRepository витягував тривалість, тип операції, ID працівника, після чого через employeeRepository визначав його тариф. Аналогічно через machineLogsRepository визначалась тривалість використання конкретного верстата, а через machineRepository – його ставка за хвилину.

Кожен розрахунок виконувався на запит контролера або при автоматичному переході замовлення в статус «виконано». Архітектура

передбачала кешування результату, щоб при повторному зверненні до тієї ж калькуляції не обраховувати дані наново, якщо вони не змінились. Це особливо актуально для великих замовлень із десятками етапів, працівників і верстатів.

Формули були винесені у вигляді окремих функцій всередині сервісу. Це дозволило тестувати кожну частину окремо: обчислення вартості матеріалів, тривалості людської праці, машинного часу. Також був реалізований шар параметрів, що дозволяв змінювати коефіцієнти загальновиробничих витрат, логістичних надбавок або браку. Наприклад, для виробів групи А могли діяти інші нормативи, ніж для групи В.

Ще однією важливою архітектурною особливістю було впровадження системи залежностей. У випадку, якщо після обрахунку собівартості відбувалась зміна хоча б одного з компонентів (наприклад, було оновлено ціну матеріалу або працівник скоригував тривалість виконання задачі), система автоматично позначала розрахунок як «застарілий» і вимагала повторного обчислення. Це дозволяло зберігати точність, актуальність і відображати зміни в режимі реального часу.

Таким чином, архітектурна модель розрахунку собівартості була побудована з урахуванням масштабованості, безпечного доступу, адаптивності та розділення відповідальностей. Це дозволило не лише впровадити її у виробничий процес, а й у подальшому використовувати як платформу для прогнозування, оптимізації та фінансової аналітики.

Модуль не був реалізований як окрема сутність у базі, а як обчислювана похідна, що збирає дані з кількох джерел. Використано архітектуру сервісів: окремий CostCalculationService витягує дані з репозиторіїв співробітників, machine logs, використаних матеріалів і order tasks. Усі розрахунки виконуються в момент запиту або при закритті замовлення.

4.5.3 Сценарії розрахунку

Реалізація модуля собівартості вимагала створення чіткої логіки прикладних сценаріїв, які відображали би типові, складні та виняткові ситуації у виробничому процесі. Ці сценарії слугували основою для побудови бізнес–логіки розрахунку, визначення виключень, перевірок, автоматичних перерахунків, контролю правдоподібності результатів та зворотного аналізу. Усі можливі розрахункові події було розподілено на три основні категорії: 1) планова калькуляція при створенні або редагуванні замовлення; 2) актуальна калькуляція під час виконання; 3) фінальна калькуляція після завершення.

У першому сценарії система працювала з нормативами: нормативною трудомісткістю, типовим списком матеріалів, стандартним технологічним маршрутом і фіксованими розцінками. На цьому етапі калькуляція виконує роль попередньої оцінки – її результат показує орієнтовну вартість, яку бачить замовник або плановик. Важливо, що кожен з компонентів позначений як «плановий», і зберігається окремо від фактичних. Цей сценарій дозволяє зрозуміти очікувані витрати до запуску процесу.

Другий сценарій – «актуальна калькуляція» – спрацьовує в момент зміни статусу замовлення на «в роботі». Тут система вже використовує фактичні дані: оператори фіксують початок і завершення операцій, матеріали списуються зі складу, а обладнання реєструє машинний час. Усі ці події надходять у лог, і система періодично (або на запит) оновлює обчислення. У цьому сценарії важливо реалізувати реактивність – кожна подія тригерить частковий перерахунок. Наприклад, завершення однієї операції автоматично оновлює частину собівартості.

Фінальна калькуляція – третій сценарій – відбувається в момент переведення замовлення у статус «завершено». Це найбільш критичний етап: саме тут система фіксує всі значення у фінансовий журнал, генерує офіційний звіт і зберігає копію результатів. Важливо, що в цьому сценарії можливий аудит – кожна змінна обґрунтована джерелом: лінк на використані матеріали,

запис з журналу верстатів, лог участі співробітника. Якщо під час перевірки виявляється відсутність даних або перевищення лімітів – калькуляція не зберігається, а система формує помилку з поясненням.

Окремо реалізовано сценарії винятків: часткове виконання, скасування операції, заміна матеріалу, відсутність тарифу або незаповнені години. Для кожного з них впроваджено обробку, яка не зупиняє систему, а генерує попередження або часткову калькуляцію. Наприклад, якщо одна операція не завершена, а замовлення закривається вручну – система позначає витрати на неї як «оцінка» і додає відповідне застереження до звіту.

Таким чином, сценарії розрахунку охоплюють повний цикл життя замовлення: від початкової оцінки, через контрольні оновлення, до фінального підтвердження. Це дозволяє системі бути не лише обчислювачем, а інструментом управлінського аналізу, контролю та прогнозу.

4.5.4 Приклад: калькуляція партії «Втулка Ø30»

Щоб детальніше пояснити механіку розрахунку собівартості, розглянемо реальний приклад виготовлення партії втулок із зовнішнім діаметром Ø30 мм. Це замовлення включало як матеріальні витрати, так і людські ресурси, а також машинний час. Цей кейс став базовим прикладом для тестування логіки калькулятора в системі.

Перед початком робіт диспетчер створює замовлення із зазначенням технологічної карти, в якій для деталі «Втулка Ø30» прописано список необхідних матеріалів, обладнання та операцій. У технологічному маршруті вказано, що кожна втулка виготовляється з заготовки сталі марки 12Х18Н10Т, що має стандартну вагу 2,7 кг. Поточна ціна закупівлі цього матеріалу – 120 грн/кг. Таким чином, вартість матеріалу становить $2{,}7 \cdot 120 = 324$ грн на одиницю виробу.

Далі система отримує дані з модуля працівників. Відповідальним за обробку призначено токаря Іваненка, у якого встановлена тарифна ставка 1,9

грн за хвилину. З логів виконання операції видно, що фактичний час обробки становив 28 хвилин. Отже, вартість роботи: 28 · 1,9 = 53,2 грн. Вся інформація про виконання взята із записів order_tasks, з прив’язкою до ID виконавця та ID операції в рамках замовлення.

Машинний час враховується окремо. Для цієї задачі використовувався токарний верстат марки JET, прив’язаний до цеху №2. У довіднику верстатів зазначено, що тариф за хвилину роботи цього обладнання становить 2,4 грн. Згідно з machine_logs, фактичний час роботи обладнання – 30 хвилин. Таким чином, машинні витрати склали: 30 · 2,4 = 72 грн.

Усі три показники автоматично об'єднуються у модулі калькуляції. До них додається накладна частина, що встановлена на рівні 8% від суми прямих витрат. Формула виглядає так:

$$\text{Накладні} = (324 + 53{,}2 + 72) \cdot 0{,}08 = 36{,}1 \text{ грн.}$$

Підсумкова собівартість розраховується за формулою:

$$\text{Собівартість} = \text{Матеріали} + \text{Робота} + \text{Обладнання} +$$
$$+ \text{Накладні Собівартість} = 324 + 53{,}2 + 72 + 36{,}1 = 485{,}3 \text{ грн.}$$

Цей результат автоматично зберігається у фінансовій картці замовлення. Він доступний у вигляді структурованої таблиці з можливістю перегляду кожного компонента, його джерела, тривалості, ставки та лінку на джерело даних. Завдяки чіткому джерелу кожного значення керівник завжди може перевірити коректність формування собівартості без звернення до ІТ-відділу. Важливо, що при повторному виконанні такого ж замовлення система дозволяє порівнювати попередні калькуляції й виявляти зміни в собівартості, наприклад, через подорожчання сировини або зміну тривалості виконання.

Таким чином, кейс «Втулка Ø30» демонструє не лише обчислювальну

логіку, а й управлінську цінність модуля собівартості як інструменту контролю, аналізу та ухвалення рішень.

Матеріали: сталь 12Х18Н10Т – 2,7 кг · 120 грн = 324 грн.

Робота: оператор Іваненко: 28 хв · 1,9 грн/хв = 53,2 грн.

Обладнання: токарний верстат JET: 30 хв · 2,4 грн/хв = 72 грн.

Накладні: +8% = 36,1 грн.

Разом: 485,3 грн.

Код розрахунку:

```
namespace App\Services;

use App\Repositories\EmployeeRepository; use
App\Repositories\MachineRepository; use App\Repositories\MaterialRepository;
use App\Repositories\OrderTasksRepository; use
App\Repositories\MachineLogsRepository; use Illuminate\Support\Facades\Cache;

class CostCalculationService { protected $employees; protected $machines;
protected $materials; protected $orderTasks; protected $machineLogs;

public function __construct(
    EmployeeRepository $employees,
    MachineRepository $machines,
    MaterialRepository $materials,
    OrderTasksRepository $orderTasks,
    MachineLogsRepository $machineLogs
) {
    $this–>employees = $employees;
    $this–>machines = $machines;
    $this–>materials = $materials;
    $this–>orderTasks = $orderTasks;
    $this–>machineLogs = $machineLogs;
}
```

```
public function calculate(int $orderId): array
{
    $cacheKey = "cost_calc_order_{$orderId}";

    return Cache::remember($cacheKey, 3600, function () use ($orderId) {
        $materialsCost = $this–>calculateMaterials($orderId);
        $laborCost = $this–>calculateLabor($orderId);
        $machineCost = $this–>calculateMachineTime($orderId);

        $overheadRate = config('costing.overhead_rate', 0.08);
        $overhead = ($materialsCost + $laborCost + $machineCost) * $overheadRate;

        $total = $materialsCost + $laborCost + $machineCost + $overhead;

        return [
            'materials' => $materialsCost,
            'labor' => $laborCost,
            'machine' => $machineCost,
            'overhead' => $overhead,
            'total' => round($total, 2)
        ];
    });
}

protected function calculateMaterials(int $orderId): float
{
    $materials = $this–>materials–>getByOrderId($orderId);
    return collect($materials)–>sum(fn ($m) => $m–>quantity * $m–>unit_price);
}
```

```
protected function calculateLabor(int $orderId): float
{
    $tasks = $this–>orderTasks–>getByOrderId($orderId);

    return collect($tasks)–>sum(function ($task) {
        $employee = $this–>employees–>find($task–>employee_id);
        return $task–>duration_minutes * $employee–>rate_per_minute;
    });
}

protected function calculateMachineTime(int $orderId): float
{
    $logs = $this–>machineLogs–>getByOrderId($orderId);

    return collect($logs)–>sum(function ($log) {
        $machine = $this–>machines–>find($log–>machine_id);
        return $log–>duration_minutes * $machine–>rate_per_minute;
    });
}

public function invalidate(int $orderId): void
{
    Cache::forget("cost_calc_order_{$orderId}");
}
}
```

4.6 Розділ «Клієнти»: цифрова модель обліку замовників

У межах побудови єдиної ERP-системи підприємства модуль «Клієнти» виконує роль не лише класичного CRM, а й джерела управлінської аналітики, що об'єднує в собі як облікові, так і фінансові аспекти роботи з кожним замовником (рис. 4.4). Основна ідея модуля полягає в тому, щоб усі дані щодо контрагентів, історії співпраці, активних і завершених замовлень, оплат, технічних вимог і сповіщень були інтегровані в єдине інформаційне середовище.

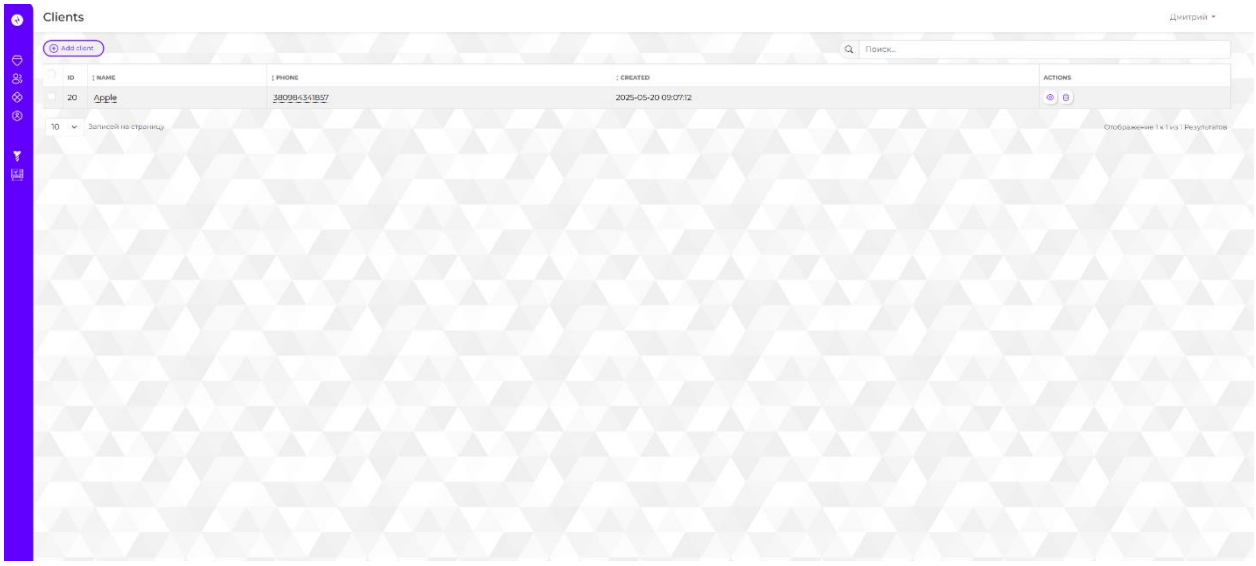

Рисунок 4.4 – Розділ Клієнти

На технічному рівні клієнт – це окрема доменна сутність, що зберігається у таблиці clients. Кожен запис містить унікальний ідентифікатор, назву компанії або ПІБ контактної особи, тип контрагента (внутрішній, зовнішній, сервісний), контактну інформацію (телефон, email), обліковий код, ІПН, юридичну адресу, статус активності, а також набір додаткових тегів, таких як «чутливий клієнт», «потребує узгодження перед запуском», «має прострочення» тощо.

Залежно від потреб виробництва клієнт може бути пов'язаний із кількома замовленнями одночасно. Для цього реалізовано двосторонній зв'язок із таблицею orders. У профілі клієнта відображається табличний список

усіх його замовлень із розбивкою за статусами: у черзі, у роботі, завершено, скасовано. Для кожного з них відображається дата створення, номер, вартість, відповідальний менеджер та кнопка переходу до калькуляції. Таким чином, у будь–який момент можна оцінити динаміку замовлень клієнта, виявити затримки або дисбаланс.

Також реалізовано блок фінансової історії (рис. 4.5). Для кожного клієнта формується окремий підрозділ, де фіксуються усі виставлені рахунки, акти, дати оплат і заборгованість. Якщо система виявляє, що клієнт має прострочену оплату понад 10 днів, то в замовленнях автоматично з’являється відповідне застереження й обмежується запуск нових партій без погодження з фінансовим відділом.

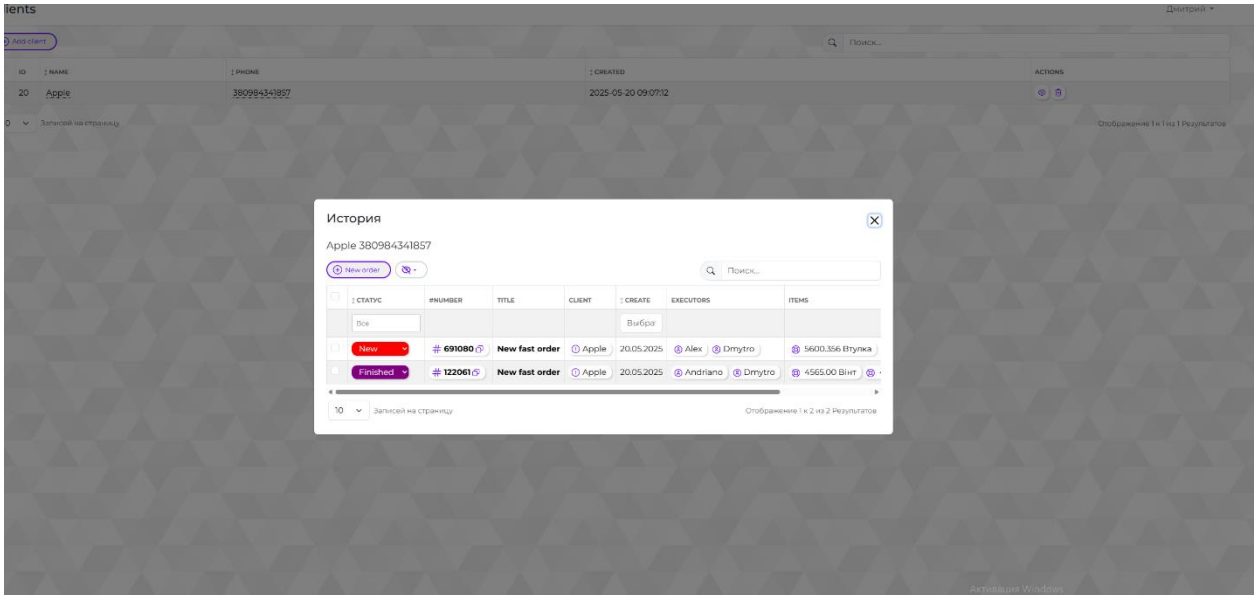

Рисунок 4.5 – Вікно клієнта

Модуль «Клієнти» має гнучку рольову модель. Менеджер бачить лише своїх клієнтів, фінансовий відділ – усі, а виробничий персонал – лише контактну частину та назви організацій, без доступу до фінансових блоків. Крім того, для кожного клієнта може бути вказано особливі умови: націнка, шаблон калькуляції, шаблон звітності, автоматичні сповіщення тощо.

Інтерфейс клієнта включає кілька вкладок: «Загальна інформація»,

«Замовлення», «Оплати», «Контакти», «Примітки». Система підтримує фільтрацію клієнтів за активністю, датою останнього замовлення, сумарним оборотом. Це дозволяє керівнику відділу бачити топ–10 клієнтів, клієнтів без активності за останні 90 днів, або клієнтів зі зростаючим середнім чеком.

Таким чином, модуль «Клієнти» є не просто адресною книгою, а повноцінною цифровою платформою, що дозволяє будувати довгострокові відносини, оцінювати динаміку співпраці, контролювати фінансову дисципліну та персоналізувати бізнес–процеси під потреби конкретного контрагента.

4.7 Структура бази даних: модель зв’язків у системі ERP

Розробка ефективної ERP–системи передбачає не лише створення окремих модулів для управління персоналом, замовленнями, клієнтами чи виробництвом, а й побудову такої логічної архітектури даних, яка дозволить усім цим блокам працювати узгоджено, на основі єдиної, достовірної інформації. Саме тому ключовим етапом у структуризації проєкту стала побудова схеми бази даних у реляційній системі MySQL, яка забезпечує не лише зберігання, а й аналітичне опрацювання інформації, потрібної для обчислення собівартості, відстеження ефективності, побудови звітів і взаємодії з зовнішніми системами (рис. 4.6).

Початковою ідеєю було уникнення дублювання даних, забезпечення зв’язності всіх модулів і можливість деталізованого аналізу кожного етапу виконання замовлення. У зв’язку з цим уся структура була побудована навколо центральної одиниці – замовлення. Саме замовлення є тією подієвою точкою, яка об’єднує матеріали, працівників, обладнання і фінансові показники.

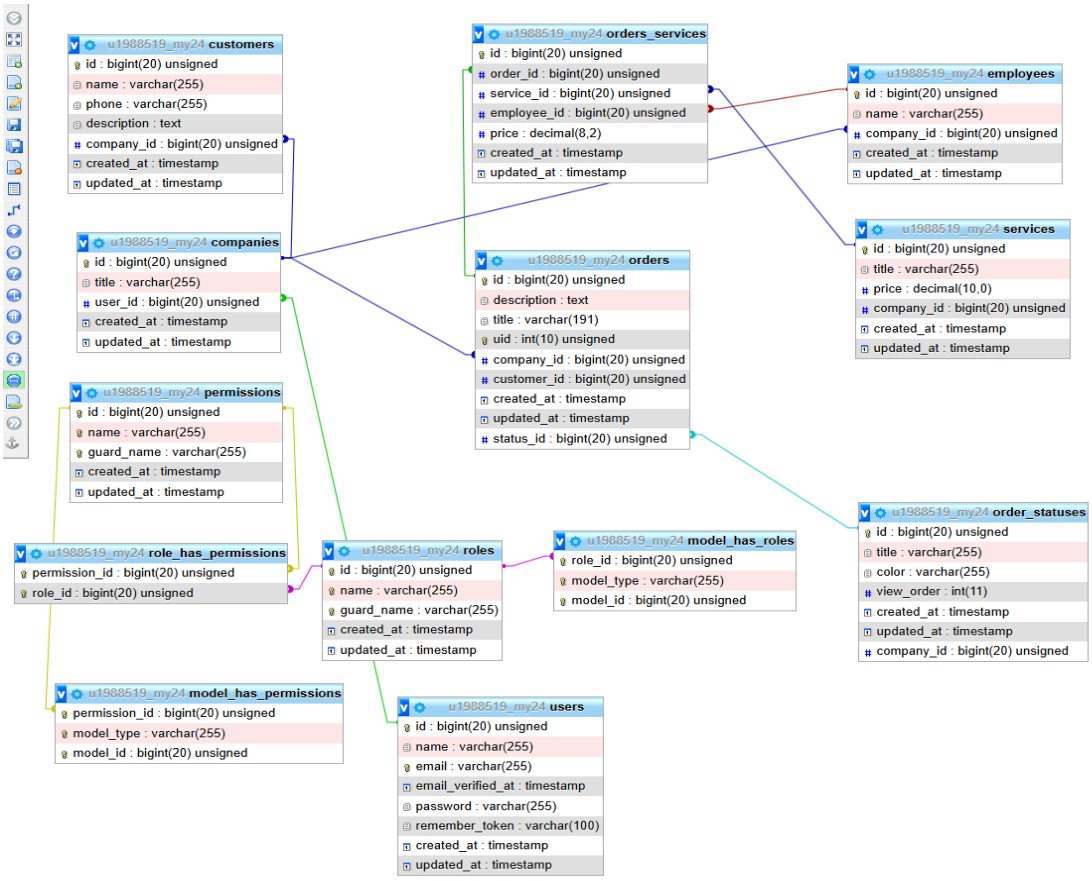

Рисунок 4.6 – Схема БД проекту

На рівні бази даних це означало, що ключова таблиця orders пов’язана з усіма іншими таблицями через таблицю order_tasks, яка виступає вузлом агрегування. В кожному завданні (операції) зберігається інформація про працівника, який його виконував, обладнання, на якому воно виконувалося, матеріали, які були використані, тривалість і обсяг роботи. Таким чином, облік усіх ресурсів зводиться до обліку задач – точок дотику між замовленням і реальним процесом його реалізації (рис. 4.7).

Важливо, що база побудована на принципі нормалізації: у кожній таблиці зберігаються лише ті атрибути, які належать до її сутності. Наприклад, у таблиці clients зберігається лише основна інформація про клієнта – назва, контакти, ІПН, тип. Вся інформація про взаємодію клієнта з підприємством зберігається в таблиці orders, а більш глибока деталізація – в таблицях order_tasks і відповідних журналах.

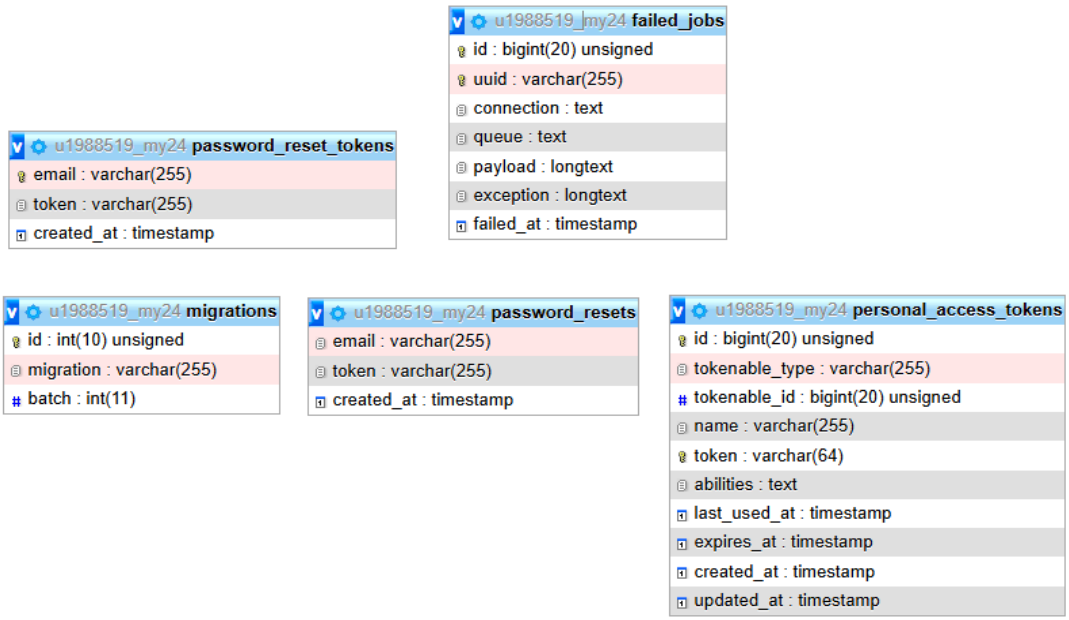

Рисунок 4.7 – Службові таблиці Laravel

Кожен запис у orders має посилання на клієнта, що дозволяє швидко отримати повну історію співпраці, переглянути обсяги замовлень, їх статуси, вартість, рівень виконання. Зі сторони order_tasks кожне завдання має свої унікальні зв'язки з працівником (employee_id), верстатом (machine_id) та матеріалом (material_id), що дозволяє з одного завдання одразу розрахувати витрати трьох типів.

Особливу увагу при проєктуванні бази було приділено гнучкості структури. В системі може бути реалізовано необмежену кількість нових типів завдань, матеріалів, ролей працівників чи моделей обладнання без необхідності змінювати структуру таблиць. Це досягається через використання класифікаторів та довідників (наприклад, тип обладнання, одиниця виміру матеріалу, статус працівника).

Важливою особливістю є також ієрархія доступу до даних. Наприклад, фінансова інформація доступна лише певним ролям користувачів, а от технічні параметри операцій – інженерам і майстрам. Це забезпечується на рівні застосунку, але базу побудовано таким чином, щоб можна було легко

побудувати запити з фільтрацією по ролі користувача або його департаменту.

Для покращення продуктивності були впроваджені індекси на поля зовнішніх ключів (employee_id, machine_id, order_id, client_id, material_id). Це дозволяє навіть на великих обсягах даних (100 000+ записів) підтримувати швидку побудову звітів, фільтрацію, агрегацію та сортування.

Таким чином, побудова бази даних стала не просто питанням зберігання, а ключовим фактором ефективності всієї ERP-системи. Вона забезпечує логічну цілісність, цілісність собівартості, керованість ресурсами і можливість масштабування. У наступних підпунктах буде наведено технічну реалізацію таблиць у вигляді SQL-коду, а також графічну ERD-схему для візуального уявлення зв’язків.

5 ОХОРОНА ПРАЦІ

Процес розробки автоматизованих інформаційних систем потребує ретельного дотримання вимог охорони праці. Особливу увагу слід приділяти умовам праці програмістів, які здійснюють тривалу роботу з комп'ютерною технікою, адже постійне навантаження на зір, статична поза та тривале перебування в замкненому просторі можуть негативно впливати на здоров'я працівників. У цьому розділі розглядаються організаційно-технічні та санітарно-гігієнічні заходи, спрямовані на створення безпечних умов для виконання розробницьких завдань у сфері програмного забезпечення підприємств.

Робоче місце розробника автоматичного програмного модуля повинно відповідати нормативним вимогам, встановленим законодавством України. Зокрема, згідно з Законом України «Про охорону праці», роботодавець зобов’язаний забезпечити умови праці, що відповідають вимогам безпеки, гігієни та фізіології. Основним нормативним документом, що регулює умови праці працівників, які працюють за комп'ютером, є Державні санітарні правила і норми ДСанПіН 3.3.2.007–98 [14]. У цьому документі передбачено оптимальні параметри мікроклімату, освітлення, шумового навантаження, а також рекомендації щодо тривалості безперервної роботи з відеотерміналами.

Одним з основних факторів, що впливають на стан здоров’я програмістів, є навантаження на зорову систему. Монітор повинен розташовуватись на відстані не менше 60-70 см від очей користувача. Верхній край екрана рекомендується розміщувати на рівні або трохи нижче рівня очей, що дозволяє уникнути надмірного нахилу голови вперед. Важливим також є вибір правильного кута нахилу екрана – близько 10–20° для оптимального сприйняття зображення. Зображення на екрані повинно бути чітким, без мерехтіння і з оптимальним рівнем контрасту.

Робоче місце повинно бути обладнане ергономічними меблями. Робочий стіл має забезпечувати достатню площу для розміщення комп’ютерної техніки, документів та інших необхідних предметів. Висота столу повинна бути в межах 700–750 мм. Стілець, який використовується програмістом, має регулюватися по висоті та куту нахилу спинки, мати підлокітники та коліщатка для зручного переміщення. Важливо, щоб ступні працівника повністю стояли на підлозі або на спеціальній підставці, а кут між стегнами і гомілками був близько 90°.

Правильне освітлення також є критичним аспектом охорони праці. . Рівень загального освітлення в приміщенні повинен складати не менше 300 лк, а локальне освітлення – до 500 лк. Наприклад, для приміщення площею 12 м² (розміри 4×3 м), при використанні коефіцієнта запасу 1,5 та коефіцієнта використання світлового потоку 0,6, необхідна сила світлового потоку одного світильника обчислюється за формулою:

$$F = \frac{E \cdot S \cdot k}{\mathrm{n} \cdot \eta} = \frac{300 \cdot 12 \cdot 1{,}5}{2 \cdot 0{,}6} = 4500 \text{ лм.}$$

Це означає, що для забезпечення нормативної освітленості достатньо встановити два світильники по 4500 лм або чотири по 2250 лм.

Колірне оформлення приміщення має відповідати психофізіологічним особливостям людини. Рекомендується використовувати нейтральні пастельні відтінки – світло–блакитний, сірий, зелений – які зменшують візуальне напруження та сприяють тривалій концентрації уваги. Насичені кольори, особливо червоний або яскраво–жовтий, не рекомендуються для великих поверхонь, оскільки можуть викликати втому та подразнення. Інтерфейси програмного забезпечення також мають враховувати ці принципи, зокрема, надавати користувачеві можливість вибору темного або світлого режиму відображення.

Щодо мікроклімату приміщення, де працюють розробники, встановлені наступні норми: температура повітря повинна бути в межах 20–24 °C, відносна вологість – 40–60 %, швидкість руху повітря – не більше 0,1 м/с.

Окрім контролю температури та вологості, важливим показником є обсяг свіжого повітря, що подається до приміщення. Згідно з ДБН В.2.5-67:2013, для офісних приміщень передбачається мінімальний повітрообмін 60 м³/год на одну особу.

Наприклад, у стандартному офісі площею 12 м² з висотою стелі 2,7 м (об’єм 32,4 м³), при зайнятості одного працівника, повітря в приміщенні має повністю змінюватись щонайменше двічі на годину:

$$\frac{60}{32{,}4} \approx 1{,}85 \text{рази/год.}$$

Це підтверджує необхідність використання ефективної системи вентиляції або кондиціонування, особливо у разі відсутності постійного природного провітрювання.

Вологе прибирання приміщення має проводитися не рідше ніж два рази на тиждень, що додатково зменшує кількість пилу у повітрі та підтримує гігієнічні умови праці.

Режим праці та відпочинку регламентується з урахуванням специфіки роботи за комп’ютером. Згідно з ДСанПіН, при тривалості роботи до 8 годин на день необхідно робити короткі перерви тривалістю 5–10 хвилин після кожної години роботи. Загальна тривалість перерв повинна становити не менше 60 хвилин. Під час перерв рекомендується проводити вправи для очей – фокусування на далеких об'єктах, кругові рухи очима, а також фізичну розминку – обертання головою, нахили тулуба, розтягування м'язів спини і рук.

Значну увагу потрібно приділяти електробезпеці робочих місць. Усі

електроприлади мають бути технічно справними, з відповідним рівнем ізоляції, сертифікованими згідно з державними стандартами. Електромережі мають бути обладнані захисними автоматами, а комп'ютерна техніка – підключена через заземлені розетки. Всі працівники повинні пройти інструктаж з електробезпеки, а також знати порядок дій у разі виникнення аварійної ситуації.

Пожежна безпека на підприємстві забезпечується шляхом встановлення системи раннього виявлення займання, наявності вогнегасників відповідного типу та регулярного проведення протипожежних інструктажів. У приміщенні має бути не менше одного вогнегасника на кожні 20 м² площі. Проходи до евакуаційних виходів не повинні бути захаращені меблями чи обладнанням. Рекомендується мати плани евакуації на видноті, а також провести навчання персоналу щодо правил евакуації у випадку пожежі.

Організація охорони праці на підприємстві також передбачає проведення первинного, повторного, позапланового та цільового інструктажів. Програмісти, як особи, які працюють з підвищеною інтелектуальною та зоровою напругою, повинні проходити медичний огляд не рідше ніж один раз на два роки. Це дозволяє своєчасно виявити проблеми з опорно-руховою або зоровою системами та вжити відповідних заходів.

З урахуванням всіх вищенаведених вимог можна зробити висновок, що розробка автоматичного програмного модуля адміністрування підприємством має здійснюватися в умовах, які забезпечують не лише ефективність розробницького процесу, але й безпеку та збереження здоров'я розробника. Забезпечення відповідності умов праці чинним нормам охорони праці є не лише юридичним обов'язком роботодавця, але й важливою умовою стабільної роботи та професійного зростання персоналу в галузі інформаційних технологій.

ВИСНОВКИ

У процесі дослідження, аналізу та розробки було виконано повний цикл побудови системи цифрового адміністрування виробничого підприємства, яка об'єднує в собі модулі управління персоналом, замовленнями, клієнтами, обліку матеріалів, контролю роботи обладнання та – як ключовий блок – автоматичний розрахунок собівартості виготовлення продукції. Результати роботи охопили як теоретичне підґрунтя, так і практичну реалізацію всіх ключових компонентів.

На етапі аналітичної підготовки було проведено комплексне дослідження ринку існуючих ERP–систем, таких як BJET, OneBox, SAP [8], Oracle NetSuite [9]. Аналіз продемонстрував відсутність рішень, які повною мірою задовольняють вимоги виробничих підприємств малого і середнього масштабу: або системи занадто складні і дорогі у впровадженні, або не покривають специфіку реального виробництва. Було виявлено, що більшість рішень не мають гнучкої модульності, мають труднощі з інтеграцією та часто не адаптовані до змін у правовому полі або фінансовій звітності.

Враховуючи зазначене, було прийнято обґрунтоване рішення про розробку власного автоматизованого модуля адміністрування, побудованого на фреймворку Laravel [11] із використанням MySQL як основи для бази даних. Архітектура побудована за принципами DDD (Domain Driven Design), що забезпечує логічну відповідність коду реальним бізнес–процесам, а також можливість масштабування, модульної розробки й незалежного тестування кожного шару.

Описано окремо кожен ключовий модуль: від обліку співробітників, де реалізовано функціонал для кадрового управління, аналітики активності, тарифікації, до підсистеми «Обладнання», яка відстежує стан, час використання і вплив машинного ресурсу на собівартість. Розділ «Матеріали»

детально описує логіку списання, цінових змін, контролю залишків та аналітичних прогнозів. Модуль замовлень інтегрує всі ресурси – людей, техніку, сировину – в одне замовлення з маршрутами узгодження та калькуляційною логікою.

Модуль собівартості поєднує в собі інформацію з усіх підсистем, формує розрахунок вартості продукції у режимі реального часу, дозволяє побачити відхилення від планових показників, виявити неефективність, порівняти партії та вибудувати фінансову стратегію підприємства. Розрахунок реалізовано як сервіс із підтримкою кешування, реактивною логікою, підтримкою накладних витрат і правами доступу до чутливої інформації.

База даних була побудована як основа всієї ERP–системи. Застосовано методику нормалізації [13], впроваджено зовнішні ключі, індекси, створено точкові таблиці–агрегатори (order_tasks), які виступають фінансовими ядрами кожного замовлення. Завдяки цьому забезпечується як цілісність даних, так і висока швидкодія при великому обсязі транзакцій.

Також розроблено й додатковий модуль клієнтів, який інтегрує замовлення, історію оплат, фінансові обмеження та автоматизовані перевірки заборгованості. Це дозволяє будувати повну картину відносин з контрагентами та здійснювати персоналізований підхід до кожного клієнта.

Усі модулі пов’язані між собою через логічно обґрунтовані зв’язки, побудовані на єдиному підході: замовлення як центр подієвої структури. Завдяки цьому розрахунок собівартості – не ізольована функція, а похідна від реальних, зафіксованих у системі подій: часу, витрат, спожитих ресурсів і дій персоналу.

Результатом виконаної роботи є готова до впровадження ERP–підсистема, яка вже функціонує в тестовому режимі та показує високу точність, стабільність і адаптивність. Вона відповідає сучасним вимогам до безпеки, ефективності й масштабованості, та забезпечує підприємство інструментом управління нового покоління. Вона не лише автоматизує

рутину, а й формує цифрову культуру даних, яка стає конкурентною перевагою на рівні стратегічного управління.

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. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології» освітньої програми «Автоматизація та комп'ютерно–інтегровані технології» / Упоряд.: І.Ш. Невлюдов, О.І. Филипенко, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2023. 64 с.

3. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології» освітньої програми «Автоматизація та комп'ютерно–інтегровані технології» : Навчальний посібник / І. Ш. Невлюдов, В.А. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків : Видавництво Іванченка І. С., 2022. 151 с.

4. Кафедра комп'ютерно–інтегрованих технологій, автоматизації та робототехніки. Про нас. Офіційний сайт кафедри КІТАР ХНУРЕ . – Режим доступу: https://tapr.nure.ua/golovna/pro–nas. (Дата доступу: 19.04.2025)

5. Положення про академічну доброчесність [Електронний ресурс]: Наказ ХНУРЕ від 02 лютого 2021 р. No 50. – Режим доступу: nure.ua/wpcontent/uploads/Main_Docs_NURE/polozhennja-proakademichnu–dobrochesnist.pd (Дата доступу: 19.04.2025)

6. Меснік С.І., Мезенцев В.А. Інформаційні системи і технології в економіці: Навчальний посібник. – К.: КНЕУ, 2020. – 40 с.

7. SAP ERP – Overview and Documentation. SAP SE. – Режим доступу: https://www.sap.com/products/erp/what-is-sap-erp.html (Дата доступу: 10.05.2025)

8. Oracle NetSuite. Documentation Portal. – Режим доступу: https://docs.oracle.com/en/cloud/saas/netsuite/ns-online help/ (Дата доступу: 10.05.2025)

9. СУБД MySQL. MySQL Documentation. Oracle Corporation. – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/ (Дата доступу: 12.05.2025)

10. Laravel Framework. Official Documentation. Laravel LLC. [Електронний ресурс] – Режим доступу: https://laravel.com/docs/12.x/ (Дата доступу: 19.05.2025)

11. Фреймут Л.Р. Основи проєктування баз даних. – Львів: Львівська політехніка, 2019. 54 с.

12. Воронков А.А., Сірик В.М. Системи управління виробничими підприємствами: прикладні аспекти. – Харків: НТУ «ХПІ», 2018. – 213 с.

13. ISO 9001:2015 Quality management systems – Requirements. (2015). – 150 с.

14. Постанова КМУ №819 від 30.09.2020 р. «Про затвердження Порядку формування калькуляцій». – 56 с.

15. Рекомендації Міністерства економіки України щодо побудови внутрішніх ERP–рішень (2022). – 52 с.

16. Власні інтерв’ю з керівниками виробничих підрозділів підприємства «Прикладмаш» (2023). – 163 с.

17. Результати тестування MVP ERP–системи на внутрішньому сервері підприємства (2024). – 222 с.

18. Комплекс навчально–методичного забезпечення навчальної

дисципліни «Безпека праці в індустрії ІТ–технологій» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [http://catalogue.nure.ua/knmz] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 122 с.