Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики та комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтеграція ERP-рішення на базі вебсервісу для оптимізації управління ресурсами промислової компанії
(тема)

Виконав:
студент 4 курсу, групи АКТСІ-20-3
Олінкевич Я.В.
(прізвище, ініціали)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Системна інженерія
(повна назва освітньої програми)

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

Допускається до захисту

Зав. кафедри _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024 р

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 Системна інженерія

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Олінкевичу Ярославу Віталійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Інтеграція ERP-рішення на базі вебсервісу для оптимізації управління ресурсами промислової компанії

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: Інтеграція ERP-рішення на базі вебсервісу для оптимізації управління ресурсами промислової компанії. Форма діалогу: вебсервіс. Перелік програмних засобів, використовуваних в роботі: Visual Studio Code, JavaScript, Node.js, MongoDB, MongoDB Atlas , Materialize.css. Технічне обладнання: комп'ютер або ноутбук із веббраузером Google Chrome.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ; 4.2 Аналіз предметної області та постановка задачі; 4.3 Розробка вимог до системи; 4.4 Опис прийнятих проєктних рішень; 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) – xx сторінок 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	Терміни виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	22.04.2024	Виконано
2	Аналіз предметної області та постановка задачі, огляд літератури з теми кваліфікаційної роботи	23.04.2024-25.04.2024	Виконано
3	Розробка вимог до системи та моделювання клієнтської частини проєкту на мові UML	26.04.2024-27.04.2024	Виконано
4	Вибір програмного стеку для реалізації проєкту	27.04.2024-28.04.2024	Виконано
5	Розробка логічної структури даних та алгоритмів роботи системи	29.04.2024-30.04.2024	Виконано
6	Розробка вебодатку	01.05.2024-28.05.2024	Виконано
7	Тестування вебдодатку	29.05.2024	Виконано
8	Розробка Посібника користувача	30.05.2024	Виконано
9	Оформлення пояснювальної записки, програмної документації та презентаційних матеріалів	31.05.2024-06.06.2024	Виконано
10	Представлення на рецензування	07.06.2024	Виконано
11	Представлення кваліфікаційної роботи в ЕК	17.06.2024	

Дата видачі завдання 22 квітня 2024 р.

Студент _____
(підпис)

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

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував 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: 75 с., 2 табл., 52 рис., 3 дод., 15 джерел.

БАЗА ДАНИХ, ВИРОБНИЦТВО, КОРИСТУВАЧ, МАРШРУТ, МОДЕЛЬ, УПРАВЛІННЯ, ФУНКЦІЯ, ERP-СИСТЕМА

Об’єкт розробки – процес управління ресурсами промислового підприємства з існуючими бізнес-процесами.

Предмет розробки – ERP-рішення для оптимізації управління ресурсами промислового підприємства.

Мета роботи – розробка ERP-рішення на базі браузерного додатку, призначенням якого є забезпечення зручного середовища для моніторингу та управління ресурсами промислової компанії.

Методи дослідження – аналіз сучасного стану питання управління ресурсами в промисловості, аналіз існуючих ERP-рішень, моделювання логіки роботи системи за допомогою UML-діаграм, розробка алгоритмів роботи системи, розробка логічної структури даних, розробка серверної та клієнтської частин проєкту за допомогою обраного стек технологій.

У кваліфікаційній роботі розглянуто актуальні питання за темою, запропоновано рішення з оптимізації робочих процесів для сучасного виробництва. Спроектовано та розроблено програмне рішення ERP-системи на базі вебдодатку для трьох працівників підприємства та протестовано його.

Програмний продукт може широко використовуватися в сфері малого та середнього бізнесу на підприємствах, пов’язаних із виготовленням великих партій виробів під замовлення.

За результатами роботи було опубліковано тези доповіді у збірнику навчального закладу та наукову статтю у науково-технічному журналі.

ABSTRACT

Explanatory note contains 75 p., 2 tables, 52 figures, 3 appendix, 15 sources.

DATABASE, ERP-SYSTEM, FUNCTION, MANAGEMENT, MODEL, PRODUCTION, ROUTE, USER

The object of development is the process of resource management of an industrial enterprise with existing business processes.

The subject of development is an ERP solution for optimizing the resource management of an industrial enterprise.

Purpose: to develop an ERP solution based on a browser application, the purpose of which is to provide a convenient environment for monitoring and managing the resources of an industrial company.

Research methods: analysis of the current state of resource management in industry, analysis of existing ERP solutions, modeling the logic of the system using UML diagrams, development of system algorithms, development of a logical data structure, development of the server and client parts of the project using the selected technology stack.

The qualification work considers topical issues on the topic, proposes solutions to optimize work processes for modern production. An ERP software solution based on a web application for three employees of the enterprise was designed and developed and tested.

The software product can be widely used in the field of small and medium-sized businesses in enterprises involved in the manufacture of large batches of products to order.

Based on the results of the work, the abstracts of the report were published in the collection of the educational institution and a scientific article in a scientific and technical journal.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

СУБД – система управління базами даних.

CRUD – функції управління даними: створення, читання, оновлення, вилучення;

ERP – планування ресурсів виробництва;

ODB – об’єктне моделювання даних;

UML – універсальна мова моделювання;

ЗМІСТ

ВСТУП

На сьогоднішній день, з розвитком технологій та зростанням конкуренції на ринку, підприємства все більше звертають увагу на впровадження ERP-систем – систем управління підприємством. Це обумовлено необхідністю оптимізації бізнес-процесів, зростанням ефективності, підвищенням конкурентоспроможності. ERP-системи інтегрують усі ключові функції та ділянки діяльності підприємства, від фінансів та управління кадрами до виробництва та логістики, надаючи комплексний погляд на діяльність підприємства. Їх впровадження дозволяє оптимізувати ресурси, зменшувати витрати, підвищувати продуктивність та покращувати якість продукції.

Основною перевагою ERP-систем є їх здатність забезпечити єдину платформу для об'єднання різних підрозділів підприємства. Це сприяє кращій комунікації між відділами, підвищенню прозорості процесів та швидшому прийняттю управлінських рішень. ERP-системи дозволяють зменшити кількість дублюючих операцій, знизити ризик помилок та покращити контроль над усіма аспектами діяльності компанії.

Важливим аспектом є також можливість масштабування ERP-системи відповідно до зростання підприємства. Це означає, що система може бути адаптована до нових вимог і змін в структурі компанії без значних витрат на її модифікацію або заміну. Впровадження ERP-системи допомагає підприємствам залишатися гнучкими та адаптивними до ринкових умов, що особливо важливо в умовах швидко змінюваного бізнес-середовища.

Мета роботи – розробка ERP-рішення на базі браузерного додатку, призначенням якого є забезпечення зручного середовища для моніторингу та управління ресурсами промислової компанії.

Об’єкт розробки – процес управління ресурсами промислового підприємства з існуючими бізнес-процесами.

Предмет розробки – ERP-рішення для оптимізації управління ресурсами

промислового підприємства.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області та опрацювати алгоритм інтеграції ERP-системи у виробництво;

– провести аналіз існуючих ERP-рішень;

– виділити базові вимоги до розробки системи ERP;

– змоделювати логіку роботи системи;

– розробити логічну структуру даних;

– обрати доречний технологічний стек та розробити серверну й клієнтську частини проєкту;

– 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 проектуванням 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 ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасного стану питання управління ресурсами в промисловості

Сучасний динамічний характер ринкових відносин, жорстка конкурентна боротьба та зростаючі запити споживачів виводять питання оптимізації управління ресурсами промислових компаній на перший план. Цьому сприяє комплекс факторів:

– новітні технології, які постійно змінюються та вдосконалюються, що створює необхідність у постійному оновленні та модернізації обладнання та процесів;

– розширення глобальних ринків та зростання міжнародної конкуренції, що вимагає вдосконалення якості продукції та ефективності виробництва;

– постійні зміни в законодавстві та стандартах безпеки та якості, які потребують постійного вдосконалення виробничих процесів для відповідності вимогам;

– зростання екологічних стандартів та соціальна відповідальність, що ставить під сумнів традиційні методи виробництва та вимагає розробки екологічно чистих технологій та процесів;

– підвищення важливості ефективного управління ланцюгами постачання та оптимізації логістичних процесів для забезпечення швидкості та якості доставки продукції споживачам.

Враховуючи ці фактори, впровадження систем оптимального управління ресурсами стає критично важливим завданням, адже воно дозволяє значно знизити витрати на закупівлю сировини, енергоносіїв, транспортних послуг; підвищити якість продукції, яка відповідатиме очікуванням споживачів; впровадити нові технології; зміцнити конкурентоспроможність; зменшити шкоду для екології.

1.2 Оцінка ефективності впровадження ERP-системи для оптимізації управління ресурсами промислової компанії

Серед комп'ютерних систем для вирішення питання оптимізації виробничих процесів вдалим рішенням буде впровадження ERP-системи (Enterprise Resource Planning). ERP-системи – це комп'ютерні системи, створені для обробки ділових операцій організації та для сприяння комплексному та оперативному (в режимі реального часу) плануванню, виробництву та обслуговуванню клієнтів. Вони спрощують обробку даних, усуваючи дублювання та неефективність, автоматизують рутинні операції, покращують комунікацію та співпрацю між відділами, а також забезпечують точну та своєчасну інформацію для прийняття обґрунтованих стратегічних та оперативних рішень. Важливою перевагою є також можливість оптимізації управління запасами, зниження витрат, вдосконалення планування та прогнозування попиту. Приклад ERP-системи зображено на рисунку 1.1.

Рисунок 1.1 – Система управління ресурсами компанії [5]

Задачі ERP-систем можуть включати в себе: об'єднання всіх функціональних областей підприємства в єдину систему, усуваючи дублювання даних та неефективність; автоматизацію рутинних операцій, звільняючи час для більш важливих завдань; покращення комунікації та співпраці між відділами; надання точної та своєчасної інформації з усіх аспектів діяльності підприємства; підтримка аналітики та звітності для прийняття обґрунтованих стратегічних та оперативних рішень; виявлення та усунення неефективності та проблемних зон; створення єдиної бази даних для всіх ресурсів підприємства; оптимізація управління запасами; зниження надлишків та витрат; покращення планування та прогнозування попиту; підвищення прозорості та контролю над ланцюгом постачання; зниження витрат за рахунок оптимізації процесів та ресурсів; підвищення продуктивності та ефективності роботи; покращення якості продукції та послуг; збільшення частки ринку та прибутку; створення гнучкої та масштабованої системи, яка може адаптуватися до нових вимог та умов; підтримка інновацій та впровадження нових технологій; покращення реакції на зміни ринкового середовища та конкурентної ситуації.

Однак, як і будь-яка інша комп’ютеризована система, ERP має ряд певних недоліків та нюансів, на які слід звертати увагу перед інтеграцією:

– великі витрати на впровадження: реалізація систем управління ресурсами може вимагати значних інвестицій у придбання програмного забезпечення, навчання персоналу та внутрішні ресурси для впровадження та підтримки системи;

– недостатня підтримка та навчання персоналу: брак адекватної підготовки та навчання персоналу щодо використання нових систем може призвести до недосягнення повного потенціалу цих систем та зниження ефективності використання;

– суперечності з існуючими процесами: інтеграція нових систем управління ресурсами може призвести до суперечок з існуючими процесами та структурами компанії, що вимагатиме часу та зусиль для вирішення цих проблем.

Останньому пункту варто приділити особливу увагу, оскільки на сьогоднішній день існує достатньо велика кількість пропозицій ERP-систем, як

цілком готових до впровадження, так і розробки під замовлення [5]. Тому питання вибору системи, яка найбільше підходитиме тому чи іншому виробництву, стоїть дуже гостро.

Одними з найбільш успішних і широко використовуваних ERP-систем у світі є SAP ERP та Oracle ERP Cloud. Їх популярність і ефективність базуються на довгих роках розвитку, інноваціях і адаптації до потреб сучасного бізнесу.

SAP ERP, розроблена німецькою компанією SAP SE, вперше побачила світ у 1972 році. Відтоді ця система постійно вдосконалювалася і розширювалася, щоб задовольнити вимоги найрізноманітніших галузей. SAP ERP має модульну архітектуру, що дозволяє компаніям обирати тільки ті компоненти, які їм необхідні. Це робить систему надзвичайно гнучкою і масштабованою. Крім того, інтеграція між різними модулями забезпечує плавний потік інформації і оптимізацію бізнес-процесів, що знижує витрати і підвищує ефективність. Завдяки вбудованим інструментам аналітики і звітності, SAP ERP допомагає компаніям приймати обґрунтовані рішення на основі точних даних. Компанії, такі як Siemens, Coca-Cola та Bosch, використовують SAP ERP для управління своїми глобальними операціями, включаючи виробництво, фінанси та логістику. Це свідчить про високу надійність і продуктивність системи.

Oracle ERP Cloud є сучасною хмарною ERP-системою, розробленою компанією Oracle Corporation. Запущена у 2012 році, ця система швидко завоювала популярність завдяки своїй інноваційності та потужному функціоналу. Oracle ERP Cloud працює в хмарі, що забезпечує доступ до системи з будь-якого місця і значно знижує витрати на IT-інфраструктуру. Регулярні оновлення проводяться без перерв у роботі, що забезпечує безперебійний бізнес-процес. Однією з ключових переваг є інтеграція передових технологій, таких як штучний інтелект та машинне навчання, які автоматизують рутинні завдання і забезпечують точний прогноз та аналітику. Широкий функціонал системи охоплює фінансовий менеджмент, управління проектами, закупівлі, управління ланцюгами поставок та людськими ресурсами. Компанії, такі як FedEx, Office Depot та Apollo Group, використовують

Oracle ERP Cloud для оптимізації своїх фінансових процесів, управління ланцюгами поставок і підвищення загальної ефективності роботи.

1.3 Аналіз потреб бізнесу та вибір ERP-системи

Перед вибором ERP-системи необхідно провести докладний аналіз потреб бізнесу. Це включає в себе розгляд різних аспектів та факторів, які впливають на застосування оптимальної ERP-системи. Важливо спрямувати увагу на наступні пункти:

– ідентифікація функціональних потреб. Основна мета – визначення функціональних областей, які мають бути охоплені системою. Це можуть бути управління виробництвом, складським обліком, фінансами, логістикою, кадрами тощо. Важливо врахувати унікальні потреби компанії в цих областях;

– оцінка поточних бізнес-процесів. Проведення детального аналізу поточних бізнес-процесів для виявлення можливостей для їх оптимізації та автоматизації за допомогою ERP-системи. Це дозволяє ідентифікувати слабкі місця та потенційні області покращень.

– врахування майбутнього розвитку бізнесу. Важливо врахувати потенційні майбутні зміни та зростання компанії при виборі ERP-системи. Система повинна бути масштабованою та гнучкою, щоб забезпечити підтримку росту бізнесу;

– аналіз можливостей інтеграції. Якщо в компанії вже використовуються інші програмні системи, необхідно визначити можливість їх інтеграції з обраною ERP-системою. Це дозволить забезпечити зручну та ефективну роботу всіх систем;

– оцінка бюджету та вартості впровадження. Важливо оцінити витрати на впровадження та підтримку ERP-системи відповідно до бюджету компанії. Необхідно зважити на вартість ліцензій, налаштування системи, навчання персоналу та підтримку після впровадження;

– функціональність системи. Сюди входить інтеграція, модульність, відповідність нормативній базі, гнучкість та можливості конфігурації;

– безпека. Гарантування надійного захисту та збереження конфіденційності системи, відповідність вимогам державних стандартів.

1.4 Впровадження ERP-системи у виробництво

Після того, як потреби компанії було ретельно проаналізовано та обрано систему, яка задовільняє всім вимогам, потрібно переходити до наступного етапу – інтеграції системи. Інтеграція ERP-системи – це складний процес, який потребує ретельного планування, координації та співпраці з боку всіх зацікавлених сторін. Основні етапи даного процесу зображено на рисунку 1.2.

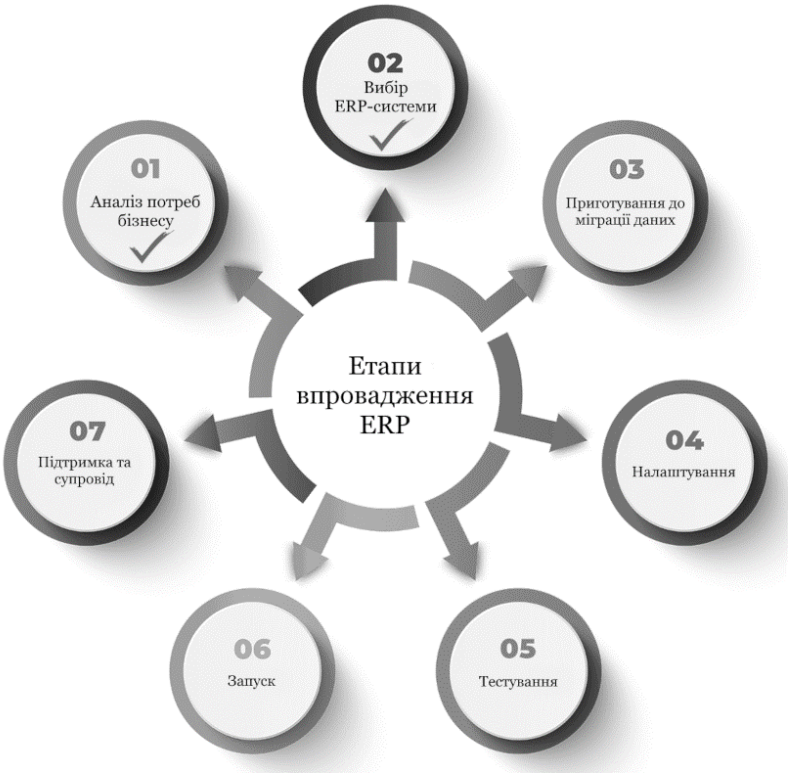

Рисунок 1.2 – Етапи впровадження ERP [5]

Міграція даних. Це може бути складним та трудомістким процесом. Важливо ретельно все спланувати, обробити та перетворити дані, щоб забезпечити їх

точність та сумісність з новою системою. Також рекомендується провести тестування міграції, щоб переконатися, що дані перенесені правильно.

Налаштування. Цей етап включає налаштування параметрів, створення користувачів та груп, визначення прав доступу, налаштування робочих процесів та інтеграцію з іншими системами, які можуть використовуватися.

Перед запуском ERP-системи в експлуатацію необхідно провести ретельне тестування. Це допоможе виявити та усунути будь-які помилки або проблеми, які можуть виникнути, а також гарантувати, що система працює правильно та відповідає очікуванням.

Деякі типи тестування, які слід провести:

– системне тестування. Процес перевірки того, чи відповідає ERP-система всім вимогам, визначеним на етапі аналізу потреб бізнесу;

– інтеграційне тестування. Перевірка коректної взаємодії ERP-системи з іншими системами, які, можливо, використовуються в компанії;

– тестування користувачів. Залучення співробітників різних відділів до тестування системи, щоб переконатися, що вона відповідає їхнім робочим потребам та інтуїтивно зрозуміла;

– тестування навантаження. Перевірка того, як система справляється з великою кількістю одночасних користувачів та транзакцій.

Після успішного завершення тестування можна запускати ERP-систему в експлуатацію. Це може бути поетапний запуск, починаючи з одного відділу або групи користувачів, а потім поступове розширення до всієї компанії.

Після запуску ERP-системи в експлуатацію знадобиться постійна підтримка та супровід. Це може включати технічну підтримку від постачальника системи, навчання та консультації для користувачів, а також оновлення та модернізацію системи для забезпечення її актуальності.

1.5 Постановка задачі

Головною задачею кваліфікаційної роботи є розробка комп’ютерної системи для управління ресурсами промислової компанії. Для того, щоб система вважалася системою ERP-типу, вона повинна вміти вирішувати основні задачі, описані в пункті 1.1.

Система поєднує в собі три компоненти: веб-сервіс для менеджерів із закупівель, веб-сервіс для менеджерів із замовлень; веб-сервіс для планувальників виробництва.

Розроблена система пропонує такі функції:

– можливість моніторингу запасів компанії;

– можливість додавати постачальників та керувати їхніми даними;

– можливість створити специфікацію для подальшого формування з неї заявки на придбання в постачальників матеріалів;

– можливість формувати інформацію про замовлення на бланку та автоматично надсилати документ на пошту партнеру;

– можливість переглядати список проведених специфікацій для замовлення матеріалів, а також аналізувати витрати компанії на закупівлю матеріалів;

– можливість додавати клієнтські замовлення до бази;

– можливість переглядати список специфікацій клієнтських замовлень, а також більш детальний список проведених специфікацій;

– можливість переглядати та редагувати список продукції для виготовлення;

– можливість переглядати звітність із продажів;

– можливість моніторити статус виготовлення замовлень;

– можливість керувати статусами виготовлення замовлень.

З переваг ERP системи можна сформулювати такі оптимізаційні критерії, які будуть враховані при розробці вебрішення:

– централізоване зберігання даних. Замість розрізнених даних у різних відділах, ERP система об'єднує всю інформацію про постачальників, клієнтів,

продажі та інші аспекти бізнесу в єдину централізовану базу. Це робить доступ до даних швидким і простим, а також гарантує їх актуальність і узгодженість.

$$k_1 = min \sum_{i=1}^{n} t_i^{callback}, i = \overline{1,n}, \quad (1.1)$$

де $t_i^{callback}$ – час відповіді системи на i-й запит користувача;

n – кількість запитів користувача;

– автоматизація бізнес-процесів. ERP система автоматизує безліч повторюваних завдань, таких як обробка замовлень, ведення обліку запасів, створення звітів та ін. Це звільняє час співробітників для більш важливої роботи, підвищує продуктивність, знижує ризик помилок і покращує загальну ефективність роботи.

$$k_2 = min \sum_{j=1}^{m} t_j^{performing\ tasks}, j = \overline{1,m}, \quad (1.2)$$

де $t_j^{performing\ tasks}$ – час виконання кожного окремого j-го завдання;

m – кількість завдань;

– поліпшення взаємодії з постачальниками та клієнтами. ERP система надає зручні інструменти для управління взаємодією з постачальниками та клієнтами. Вся інформація про контакти, угоди, запити та історію співпраці в збирається та зберігається одному місці. Це допомагає покращити комунікацію, організувати персоналізований підхід до кожного партнера, підвищити рівень обслуговування та будувати міцніші, довгострокові стосунки.

$$k_3 = max \sum_{l=1}^{t} level_l^{satisfaction}, l = \overline{1,t}, \quad (1.3)$$

де $level_l^{satisfaction}$ – рівень задоволеності l-го партнера;

t – кількість партнерів;

$$k_4 = min \sum_{l=1}^{t} level_l^{unsatisfaction},\ l = \overline{1,t}, \tag{1.4}$$

де $level_l^{unsatisfaction}$ – рівень незадоволеності l-го партнера;

t – кількість партнерів;

$$k_5 = max \frac{t^{satisfacted} - t^{critics}}{t} \tag{1.5}$$
$$t = t^{satisfacted} + t^{critics},$$

де $t^{satisfacted}$ – кількість задоволених партнерів;

$t^{critics}$ – кількість незадоволених партнерів;

– економія часу та ресурсів. ERP система економить час та ресурси за рахунок автоматизації, оптимізації процесів та усунення дублювання даних. Це призводить до скорочення витрат, підвищення рентабельності та вивільнення коштів для інвестування в розвиток бізнесу.

$$k_6 = min\left(\left(\frac{Q}{q}\right) * c^{transfer} + \sum_{i=1}^{Q/q} c_i^{storage}\right), \tag{1.6}$$

де Q – загальна кількість ресурсів необхідних на кожен період;

q – кількість ресурсів кожної партії;

$c^{transfer}$ – вартість доставки кожної партії;

$c_i^{storage}$ – вартість зберігання кожної i-ї партії.

2 РОЗРОБКА ВИМОГ ДО СИСТЕМИ

2.1 Формулювання функціональних вимог до розроблюваної системи

2.1.1 Окреслення системних вимог до компонентів системи

Всі компоненти, призначенням яких є автоматизація системи планування ресурсів підприємства, повинні відповідати певним вимогам. Системні вимоги до цих компонентів включають:

– побудову системи на основі трирівневої архітектури, яка передбачає наявність таких компонентів, як клієнтська частина, сервер застосунків та сервер бази даних;

– серверна частина системи має бути розроблена на базі платформи NoSQL-СУБД MongoDB;

– клієнтська частина повинна бути реалізована у вигляді веб-сторінок, які здатен відкрити будь-який сучасний веб-браузер, що підтримує технології HTML5, CSS3 та JavaScript;

– клієнтська частина повинна містити інтерактивні елементи веб-інтерфейсу, що дозволять користувачам взаємодіяти з системою.

2.1.2 Розробка вимог до клієнтської частини розроблюваної системи

Основною вимогою до функціоналу інтерфейсу клієнту системи є розробка трьох незалежних, але взаємопов’язаних веб-інтерфейсів: для роботи менеджера з закупівель, менеджера з замовлень та планувальника виробництва. Ці веб-сервіси мають забезпечувати користувачам доступ до відповідних функціональних можливостей, одночасно зберігаючи можливість взаємодії між ними для забезпечення автоматизації виробничих процесів, а також плавної та ефективної роботи всієї системи.

Для отримання доступу до будь-якого веб-сервісу системи уповноважений працівник підприємства має пройти процедуру автентифікації та авторизації на

окремій веб-сторінці, після чого, в залежності від його ролі, його буде перенаправлено до відповідної частини системи.

Для роботи менеджера з закупівель, менеджера з замовлень та планувальника виробництва їх користувацькі інтерфейси мають містити сторінки та форми, що підтримують функціонал, описаний в пункті 1.5.

2.2 Моделювання логіки роботи системи за допомогою мови UML

Універсальна мова моделювання (Unified Modelling Language або UML) – це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення [6].

UML дозволяє створювати візуальні моделі, які чітко та лаконічно описують компоненти системи, їх взаємозв'язки та поведінку. Ця мова ґрунтується на відкритих стандартах і є мовою широкого профілю. Це означає, що її можна використовувати для моделювання різних типів систем, не лише програмного забезпечення. UML використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю. Ця модель слугує основою для подальшого проектування, розробки та документування системи.

Окрім моделювання програмного забезпечення, UML також потрібна для:

– моделювання бізнес-процесі, щоб відображати та описувати етапи бізнес-процесів та їх взаємозв'язки;

– системного проектування, щоб розробляти загальну архітектуру складних систем, що включають програмне забезпечення, апаратне забезпечення та інші компоненти;

– відображення організаційних структур, щоб візуалізувати структуру організації, підрозділи та їх взаємозв'язки.

Ієрархію діаграм в UML 2.2 зображено на рисунку 2.1.

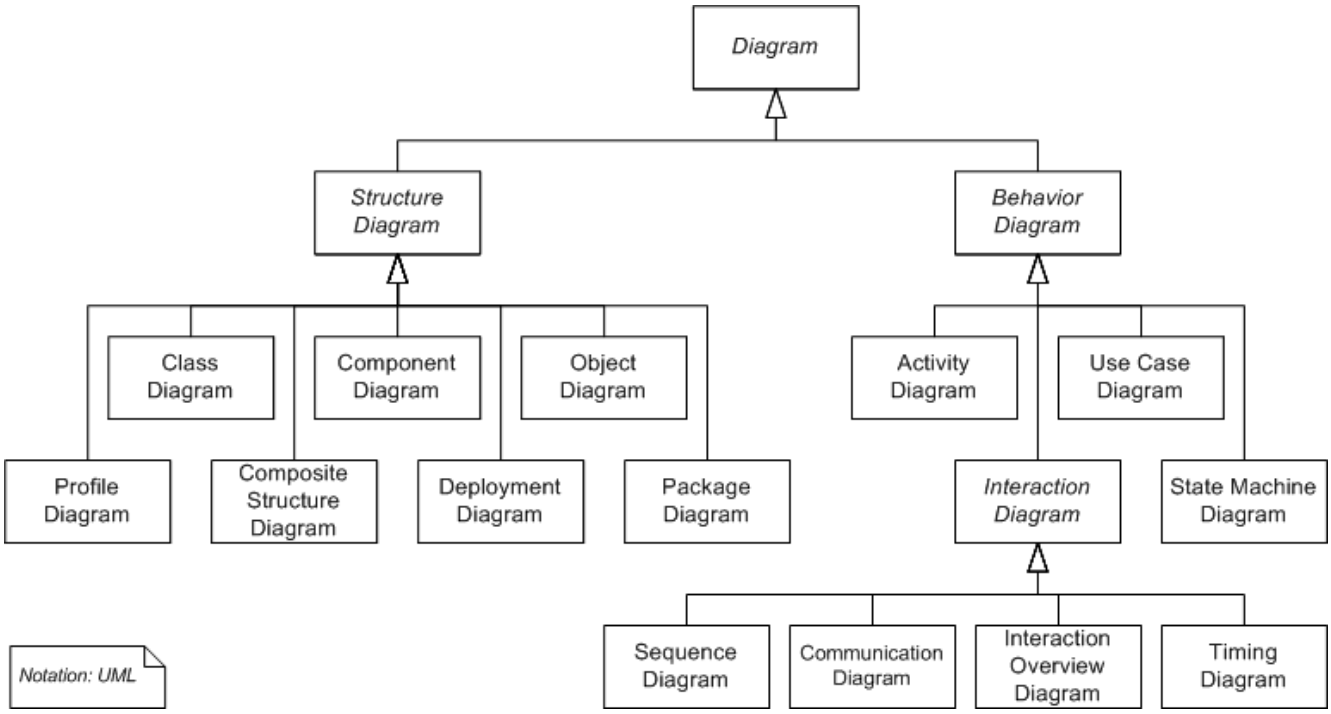

Рисунок 2.1 – Ієрархія діаграм UML 2.2 [7]

2.2.1 Розробка діаграми прецедентів для ERP-системи

Діаграми прецедентів в UML є одним із видів діаграм, що використовуються для моделювання функціональних вимог системи з точки зору користувача. Ці діаграми допомагають ідентифікувати функціональність системи та її взаємодію з користувачами або зовнішніми системами. Основні компоненти діаграм прецедентів включають:

– прецеденти, що представляють окремі функціональні можливості або дії, які може виконати система;

– акторів, тобто будь-яку роль або сутність, яка взаємодіє з системою;

– відношення, що обумовлюють взаємозв’язки між прецедентами та акторами.

В розроблювальній системі акторами є менеджер із закупівель, менеджер із замовлень та планувальник виробництва. Серед існуючих прецедентів є наступні: закупівля матеріалів, моніторинг ресурсів компанії, документування та звітність, обробка замовлень, керування робочим процесом.

Розроблена діаграма прецедентів зображена на рисунку 2.2. В таблицях 2.1 та 2.2 наведено короткі описи прецедентів та акторів.

Рисунок 2.2 – Діаграма прецедентів

Таблиця 2.1 – Описи прецедентів

Назва прецеденту	Опис
Закупівля матеріалів	Цей прецедент охоплює процес придбання необхідних матеріалів та компонентів для виробництва продукції. Включає в себе вибір оптимальних умов поставки, здійснення покупок.
Моніторинг ресурсів компанії	Забезпечує відстеження та аналіз наявних матеріалів та компонентів, необхідних для виробництва.
Документування та звітність	Цей прецедент включає в себе процес створення, збереження та обробки документів компанії, а також формування звітів для внутрішнього використання або подання стороннім організаціям.
Обробка замовлень	Цей прецедент описує процес створення та обробки замовлень клієнтів. Включає в себе прийняття замовлення, внесення його в систему, передачу планувальнику, обробку, реалізацію, моніторинг етапів

Продовження таблиці 2.1

1	2
	виготовлення, відправлення клієнтові та вчасне вирішення потенційних проблем.
Керування робочим процесом	Цей прецедент описує управління робочими процесами та завданнями в межах організації, включаючи розподіл завдань, контроль виконання та координацію між різними підрозділами.

Таблиця 2.2 – Описи акторів

Назва актора	Опис
Менеджер із закупівель	Це особа, відповідальна за придбання необхідних матеріалів та компонентів для виробництва продукції. Вона веде взаємодію з постачальниками, проводить перемовини щодо умов поставки та цін, та здійснює закупівельні операції в межах бюджету компанії.
Менеджер замовлень	Це особа, що координує процес обробки та виконання замовлень від клієнтів. Вона відстежує статус замовлень, взаємодіє з виробництвом щодо виготовлення продукції та забезпечує вчасну доставку клієнтам.
Планувальник виробництва	Це особа, яка відповідає за розподіл робочих завдань та ресурсів для оптимального виробництва продукції. Вона контролює запаси та розподіляє завдання між робочими групами.

Ключовим прецедентом, навколо якого будується система, є Обробка замовлень. Нижче наведено розгорнутий опис даного прецеденту.

Прецедент 4 Обробка замовлень.

Масштаб**:** системи виробничого управління.

Рівень**:** задача рівня користувача.

Головний актор**:** менеджер замовлень.

Зацікавлені особи та їхні вимоги:

– менеджер замовлень. Хоче точно і швидко ввести дані щодо замовлення, отримані від замовника, оскільки невірне розуміння потреб клієнта може призвести до невідповідності між очікуваннями замовника та фактично фиготовленою продукцією. Хоче знати, на якому етапі виготовлення знаходиться замовлення для відповідності плану. Хоче, щоб замовлення було виготовлено в термін, щоби вчасно відвантажити його клієнтові;

– планувальник виробництва. Хоче отримати повний опис необхідного клієнтові замовлення. Хоче мати доступ до ресурсів компанії. Хоче вчасно відвантажити готове замовлення на склад;

– клієнт. Хоче отримати замовлення, відповідаюче вказаним вимогам, належної якості та в оговорений термін;

– керівник. Хоче отримувати звітність щодо угод із клієнтами, продуктивності виробництва;

– компанія. Хоче, щоб клієнти вчасно отримували замовлення, які відповідають їхнім потребам. Хоче впевнитися, що всі вимоги клієнта виконані, обладнання справно відпрацьовувало на кожному етапі виготовлення виробів, а працівники мали доступ до всієї необхідної для роботи інформації.

Передумови**:** авторизацію менеджера замовлень проведено успішно.

Результати**:** замовлення виготовлене та відправлене клієнтові. Дані про успішну угоду з клієнтом збережено.

Основний успішний сценарій:

а) клієнт зв'язується з менеджером замовлень одним із доступних способів та надає заявку на замовлення, оговоривши умови доставки та оплати;

б) менеджер замовлень заносить дані про замовлення до бази;

в) заносячи замовлення до бази, менеджер замовлень атвматично передає його планувальнику виробництва, користувачеві, відповідному за обробку замовлення та подальше виготовлення виробу(ів);

г) менеджер замовлень отримує від планувальника виробництва інформацію щодо кожного етапу виготовлення виробу(ів);

д) після завершення виготовлення виробу(ів) та відвантаження його(їх) на склад менеджер замовлень повідомляє клієнта про готовність його замовлення та оформлює відправлення замовлення клієнтові;

е) менеджер замовлень закриває виконане замовлення.

Розширення (або альтернативні потоки):

а) При кожному виході системи з ладу (для введення системи до ладу необхідно забезпечити відновлення виконання виробничих процесів з будь-якого кроку сценарію):

– менеджер замовлень звертається до системного адміністратора для перезапуску системи;

– системний адміністратор перевіряє успішність авторизації для облікового запису менеджеру замовлень та пробує відновити попередній стан системи;

– система відновлює попередній стан.

б) Система визначає аномалію, яка спричинила збій:

– система повідомляє про помилку системного адміністратора і менеджера замовлень, реєструє помилку та переходить у початковий стан;

– менеджер замовлень відновлює виконання виробничих процесів з моменту їх зупинки.

в) Під час передачі замовлення планувальнику менеджер замовлень виявляє, що не всі необхідні дані про замовлення були введені або вони некоректні:

– менеджер замовлень повідомляє клієнта про необхідність уточнення деталей або коригує неправильно введені дані;

– менеджер замовлень знову передає замовлення планувальнику виробництва.

г) Менеджер замовлень виявляє затримку у виготовленні виробу:

– він повідомляє клієнта про затримку та пропонує альтернативні варіанти: збільшення терміну виготовлення, вибір іншого товару тощо;

– після узгодження з клієнтом менеджер замовлень вносить зміни до замовлення та інформує планувальника виробництва про необхідні корективи.

ґ) При отриманні інформації про стан виготовлення виробу(ів) виявляється, що деякі дані неактуальні:

– менеджер замовлень звертається до планувальника виробництва або системного адміністратора для вирішення проблеми.

д) Під час моніторінгу етапів виготовлення виробу(ів) менеджеру замовлень поступає інформація про технічні складнощі або великий відсоток браку:

– менеджер замовлень оцінює вплив проблеми на готовність замовлення та можливість його вчасної доставки;

– він спільно з виробничим відділом шукає шляхи вирішення проблеми, можливі альтернативні варіанти виготовлення;

– за потреби менеджер замовлень інформує клієнта про затримку та пропонує альтернативні варіанти або компенсацію за незручності;

– після вирішення проблеми або прийняття іншого рішення менеджер оновлює дані в системі та надає відповідну інформацію клієнтові.

е) При готовності замовлення менеджер замовлень виявляє проблему з доставкою через непередбачені обставини (наприклад, транспортні затори або погодні умови):

– він шукає альтернативні маршрути доставки або способи вирішення проблеми та повідомляє клієнта про затримку;

– після знаходження оптимального рішення менеджер замовлень оформлює відправлення та повідомляє клієнта про новий час доставки.

Частота використання**:** постійно.

Відкриті питання:

– як система враховує та використовує інформацію про попередні замовлення та їхні умови для оптимізації процесу виробництва та обробки нових замовлень;

– яким чином менеджер замовлень інформується щодо етапів виготовлення виробу(ів);

– чи існує процедура реагування на зміни в замовленні під час виробництва, які можуть вплинути на процес його виконання чи час доставки.

2.2.2 Розробка діаграм послідовностей та комунікації для ERP-системи

Діаграма послідовностей (Sequence Diagram):

– ця діаграма показує інформацію, що передається між об'єктами в системі під час виконання сценарію [8];

– кожен об'єкт представлений вертикальною лінією, а повідомлення між ними відображаються стрілками, які показують напрямок обміну;

– діаграма послідовностей дозволяє визначити порядок виконання операцій та зрозуміти взаємодію між об'єктами під час виконання певного сценарію.

Діаграма комунікації (Communication Diagram):

– ця діаграма здебільшого фокусується на на структурі взаємодії об’єктів;

– діаграма комунікації дозволяє показати структуру взаємодії між об'єктами та її сценарії, а також ідентифікувати об'єкти, які приймають участь у взаємодії.

Розроблена діаграма послідовностей представлена на рисунку 2.3, а діаграму комунікацій зображено на рисунку 2.4.

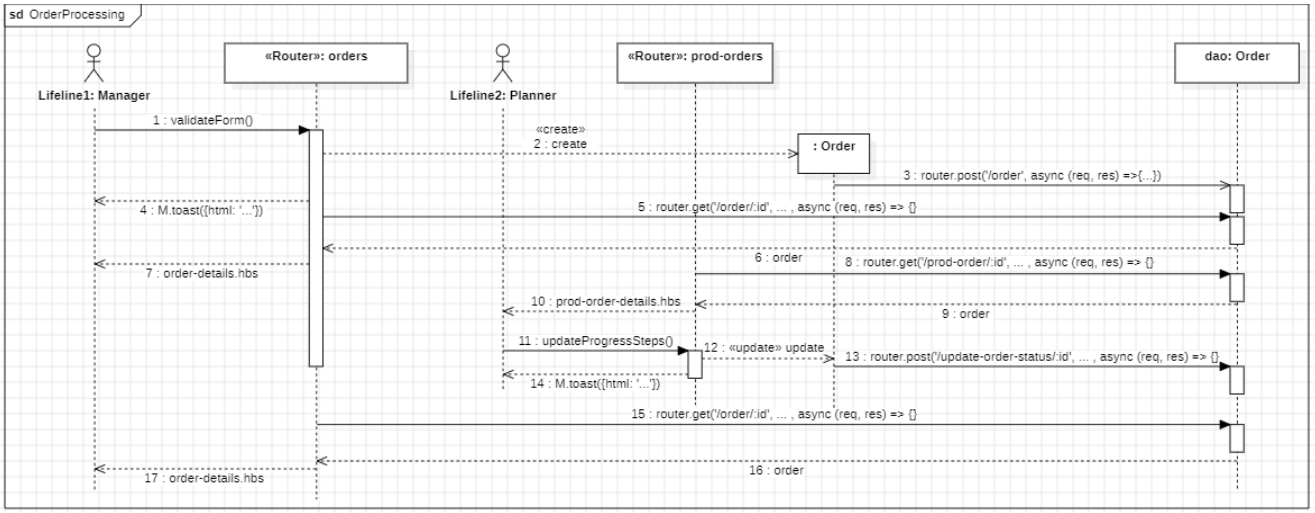

Рисунок 2.3 – Діаграма послідовностей

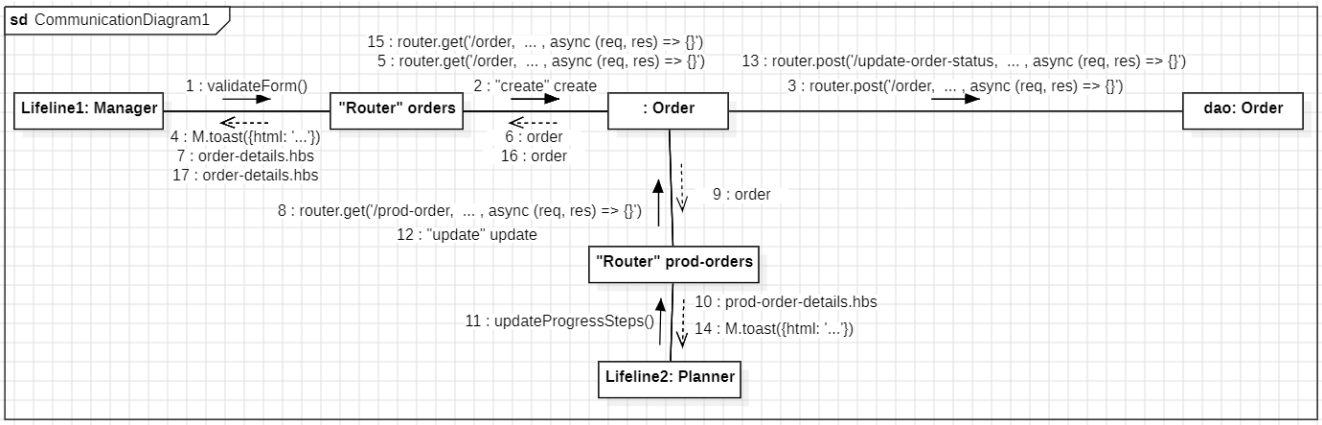

Рисунок 2.4 – Діаграма комунікацій

На діаграмах наведено взаємодію об’єктів для реалізації наступних операцій:

– validateForm() – створення менеджером замовлень нового замовлення;

– “create” create – процес створення замовлення;

– router.post(‘/order’, async(req,res) =>{…}) – POST-запит для внесення введеної менеджером інформації до бази даних;

– router.get(‘/order’, async(req,res) =>{…}) – запит до бази даних для отримання специфікації замовлення;

– order – специфікація замовлення;

– order-details.hbs – відображення інформації про специфікацію на сторінці;

– router.get(‘/prod-order’, async(req,res) =>{…}) – запит до бази даних для отримання інформації про замовлення;

– prod-order-details.hbs – відображення інформації про замовлення на сторінці;

– updateProgressSteps – оновлення планувальником статусу замовлення;

– “update” update – процес оновлення статусу замовлення;

– router.post(‘/update-order-status’, async(req,res) =>{…}) – POST-запит для внесення зміненого планувальником статусу до бази даних;

– M.toast({html: ‘…’}) – тост-повідомлення в інтерфейсі працівника;

2.2.3 Розробка діаграми класів для ERP-системи

Діаграма класів в UML – це вид діаграми, що використовується для візуалізації структури класів і взаємозв'язків між ними в системі. Вона дає нам найбільш повне і розгорнуте уявлення про зв'язки в програмному коді, функціональність та інформацію про окремі класи. Додатки генеруються часто саме з діаграми класів [9]. На діаграмі класів кожен клас відображається у вигляді прямокутника з трьома розділами: верхній містить назву класу, середній – атрибути класу, а нижній – методи класу. Відношення між класами позначаються стрілками, що вказують на тип взаємодії: асоціації, композиції, агрегації, спадкування та інші.

Створена діаграма класів представлена на рисунку 2.5.

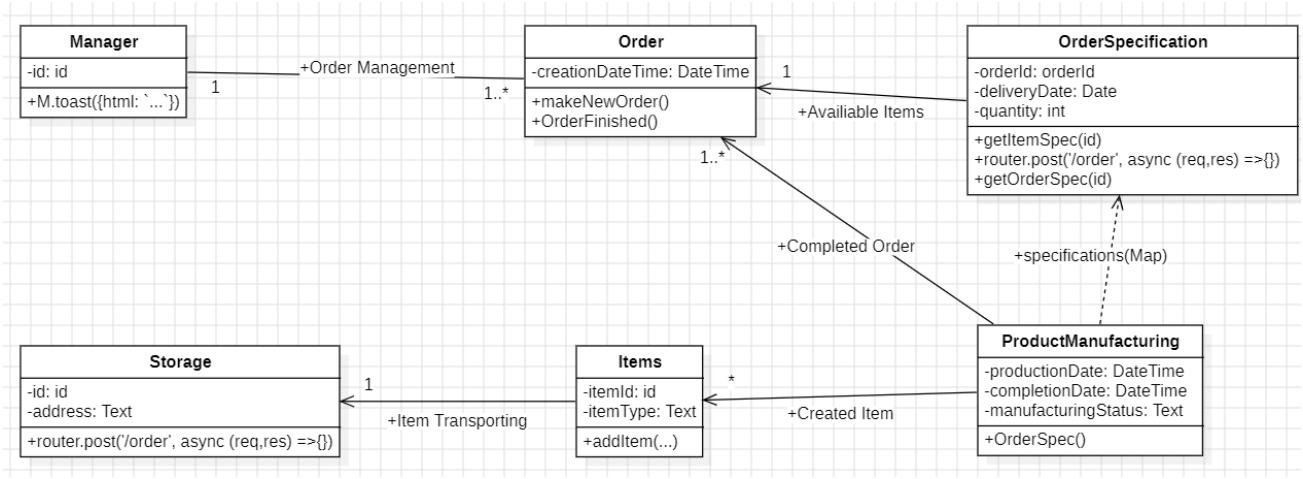

Рисунок 2.5 – Діаграма класів

2.3 Охорона праці

Офісне приміщення, що відведено для працівників, які займаються управлінням ресурсами компанії, має такі характеристики:

– площа приміщення 40 м^2 (5 м × 8 м);

– висота 4 м;

– кількість робочих місць з ПК – 6 шт.

Приміщення, відповідно до ДНАОП 0.00-1.31-99, повинно забезпечувати 6 м^2 площі і 20 м^3 об’єму на одне робоче місце з ПК. Площа приміщення складає 40 м^2, а її об'єм дорівнює 160 м^3 , на кожне місце припадає 7 м^2 площі та 26,7 м^3 об'єму. Отже, вимога виконана.

Приміщення з комп'ютерами повинні бути обладнані природним і штучним освітленням згідно з ДБН В.25-28-2006 "Природне і штучне освітлення". Природне світло має надходити через бокові вікна, орієнтовані переважно на північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не менше 1,5 %.

Рівень загального штучного освітлення приміщення можна перевірити за допомогою методу питомої потужності.

Розрахункова формула методу:

$$W = \frac{W_\Sigma}{S}, \qquad (2.1)$$

де W – питома потужність;

S – площа приміщення;

W_Σ – загальна потужність освітлювальної установки.

W_Σ розраховується за формулою:

$$W_\Sigma = W_{СВ} \cdot n_{СВ}, \qquad (2.2)$$

де $W_{св}$ – потужність одного світильника;

$n_{св}$ – кількість світильників в приміщенні.

$$W_\Sigma = 100 \cdot 6 = 600 \text{ Вт},$$

$$W_\Sigma = 600 / 40 = 15 \text{ Вт/м}^2 .$$

Питомій потужності 15 Вт/м^2 відповідає освітленість в 1024,5 Лк. при мінімальній допустимій освітленості 300 Лк. Отже, в кімнаті створені сприятливі умови за освітленням.

3 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

3.1 Обґрунтування вибору програмного стеку для реалізації проєкту

Для виконання задачі створення веб-сервісу для організації та управління ресурсами промислової компанії було обрано зв’язку з мови програмування JavaScript, платформи Node.js, фреймворка Express.js та СУБД MongoDB.

JavaScript є однією з найбільш поширених мов програмування для веб-розробки. Вона створює елементи для покращення взаємодії відвідувачів сайту з веб-сторінками, такі як випадаючі меню, анімована графіка та динамічні кольори фону [10]. Його основні переваги включають високу продуктивність, можливість розробки як фронтенд, так і бекенд частин веб-додатків, а також наявність великої спільноти розробників, що забезпечує підтримку та розвиток численних бібліотек та фреймворків.

Node.js – це обгортка для рушія JavaScript під назвою V8 від Google, яка підтримується багатьма браузерами, включаючи Google Chrome, Opera, Safari, Microsoft Edge, Firefox тощо [11]. Основні переваги Node.js включають:

– асинхронність та неблокуючу модель вводу-виводу, які дозволяють обробляти велику кількість одночасних запитів;

– використання однієї мови для серверної та клієнтської частин, що дозволяє спростити процес розробки та зменшити витрати часу на його реалізацію;

– велику екосистему модулів: завдяки npm (Node Package Manager) розробники мають доступ до тисяч модулів, які можуть бути легко інтегровані у проєкт.

Express – це мінімалістичний веб-фреймворк для Node.js, який забезпечує основні функції для створення веб-додатків і API. Переваги Express включають:

– простий та інтуїтивно зрозумілий API, що дозволяє швидко створювати сервери та маршрути;

– завдяки Middleware (проміжне ПЗ) Express можна легко налаштовувати та розширювати для задоволення конкретних потреб додатка. Існують бібліотеки для роботи з файлами cookie, сесіями, логінами користувачів, параметрами URL, POST-даними, заголовками безпеки тощо [12];

– Express не нав’язує жорсткої структури додатка, що дозволяє розробникам самостійно вибрати архітектуру та необхідні інструменти.

Для реалізації проєкту було застосовано NoSQL СУБД з відкритим вихідним кодом під назвою MongoDB. NoSQL означає, що база даних не використовує реляційні таблиці, як традиційні SQL СУБД. Існує ряд типів баз даних NoSQL, але MongoDB зберігає дані в JavaScript-подібних об'єктах, відомих як документи [13]. Основними перевагами, які вплинули на вибір саме цієї системи управління, є:

– документно-орієнтована модель, яка заберпечує високу гнучкість у моделюванні даних. Це особливо корисно для проєктів, де структура даних може змінюватися або розширюватися з часом;

– підтримка горизонтального масштабування за допомогою шардінгу (Sharding – процес збереження записів даних на декількох машинах [14]). Це важливо для великих промислових компаній, де обсяг даних може бути дуже великим, і необхідно забезпечити високу продуктивність та швидкість доступу до даних;

– автоматичне розподілення навантаження між серверами, що входять до кластеру. Це забезпечує рівномірне навантаження та запобігає перевантаженню окремих серверів, що покращує загальну надійність та продуктивність системи;

– підтримка реплікації, тобто збереження даних на декількох серверах, забезпечує безперервну роботу сервісу навіть у випадку відмови одного з серверів;

– простота інтеграції з JavaScript та Node.js завдяки офіційному драйверу. Це дозволяє легко та ефективно працювати з базою даних з використанням тих самих технологій, що і для інших частин проекту, що зменшує складність та підвищує продуктивність розробки.

3.2 Розробка алгоритмів роботи системи

Робочі області працівників підприємства містять в собі багатий функціонал, який потребує чіткої і логічної поведінки під час роботи з ними.

3.2.1 Розробка алгоритму поведінки менеджера із закупівель

Для відображення послідовності дій менеджера із закупівель при роботі з веб-сервісом розроблено алгоритм, який зображено на рисунку 3.1.

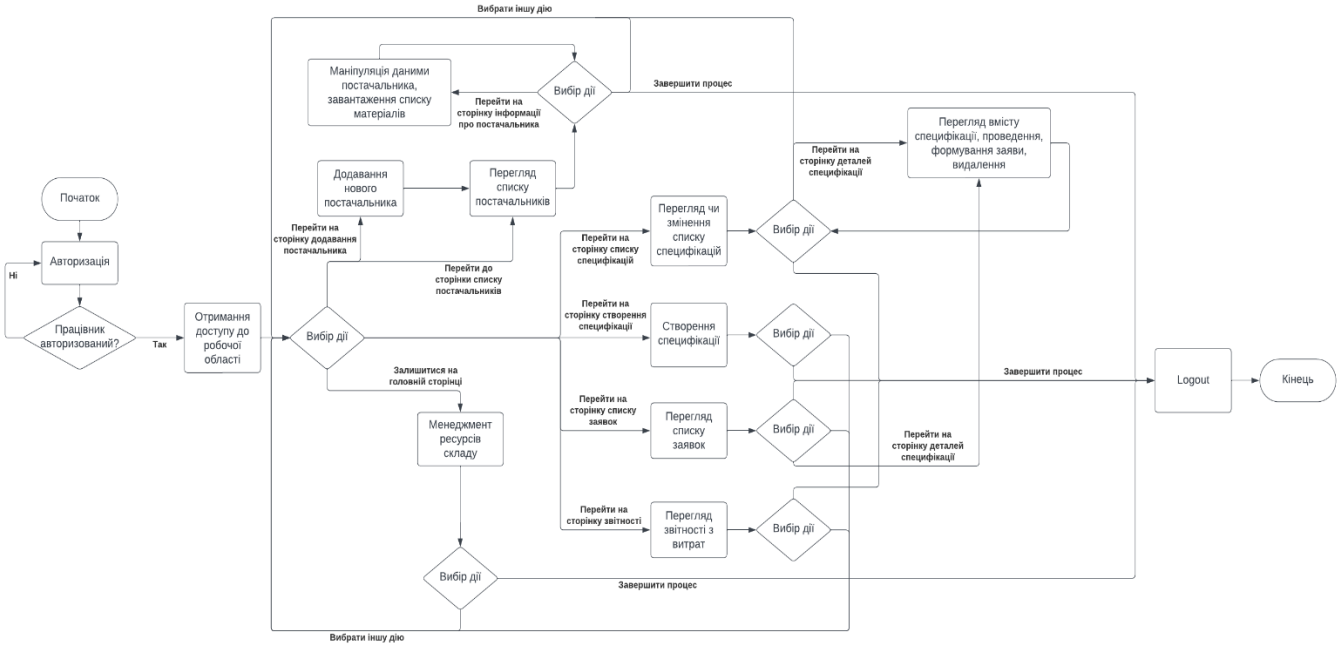

Рисунок 3.1 – Схема алгоритму послідовної поведінки менеджера із закупівель

Послідовність використання елементів веб-сервісу має наступні етапи:

– блок 1 Авторизація. Припускається, що всіх працівників реєструють в базі даних поза веб-сервісом, після чого їм надаються дані для входу в систему. Після внесення до форми входу особистого логіну та паролю користувач проходить процедуру автентифікації – перевірки на наявність користувача в системі;

– блок 2 Перевірка на наявність користувача в системі. Якщо введені дані збігаються з даними в системі, користувач отримує доступ до своєї робочої області;

– блок 3 Менеджмент ресурсів складу. Головна сторінка інтерфейсу користувача. На ній можливо стежити за ресурсами підприємства, додавати, оновлювати та видаляти дані про матеріали;

– блок 4 Додавання нового постачальника. На цій сторінці можливо занести інформацію про нового постачальника до бази для майбутньої співпраці із ним;

– блок 5 Перегляд списку постачальників. Всі наявні в базі постачальники відображаються на даній сторінці. Можливо обрати одного постачальника для відкриття детальної інформації про нього;

– блок 6 Маніпуляція даними постачальника, завантаження матеріалів. На сторінці відображається раніше внесена інформація про постачальника. Передбачено можливість автоматичного занесення списку матеріалів, отриманих від постачальника, із зовнішнього файлу до бази даних;

– блок 7 Створення специфікації. На цій сторінці можна оформити специфікацію на замовлення матеріалів у певного постачальника зі списку матеріалів, завантаженого до системи раніше. Система передбачає автоматичний розрахунок вартості всіх обраних матеріалів;

– блок 8 Перегляд чи змінення списку специфікацій. Всі сформовані специфікації відображаються в цьому списку. Є можливість видалення специфікації зі списку (але не з бази даних) та отримання детальної інформації щодо специфікації за кліком;

– блок 9 Перегляд вмісту специфікації, проведення, формування заяви, видалення. На цій сторінці працівник може переглянути створену специфікацію, провести її, щоб зафіксувати, що замовлення оформлене для передачі постачальнику, далі сформувати pdf-документ для друку або ручного відправлення постачальникові на пошту. Також присутня можливість проведення та формування pdf-документу з автоматичною відправкою на електронну адресу постачальника. Є можливість видалення специфікації із бази даних;

– блок 10 Перегляд списку заявок. На цій сторінці, на відміну від списку специфікацій, відображаються лише проведені специфікації, які тепер

ідентифікуються як заявки. Можливе відстеження всіх наявних заявок та отримання детальної інформації про заявку за кліком;

– блок 11 Перегляд звітності з витрат. Ця сторінка надає інформацію з витрат за кожен місяць кожного року, сформовану, виходячи з виконаних замовлень, та представлену у вигляді лінійної діаграми;

– блок 12 Logout. Завершення процесу роботи працівника шляхом повернення на сторінку авторизації.

3.2.2 Розробка алгоритму поведінки менеджера із замовлень

Для відображення послідовності дій менеджера із замовлень при роботі з веб-сервісом розроблено алгоритм, який зображено на рисунку 4.2.

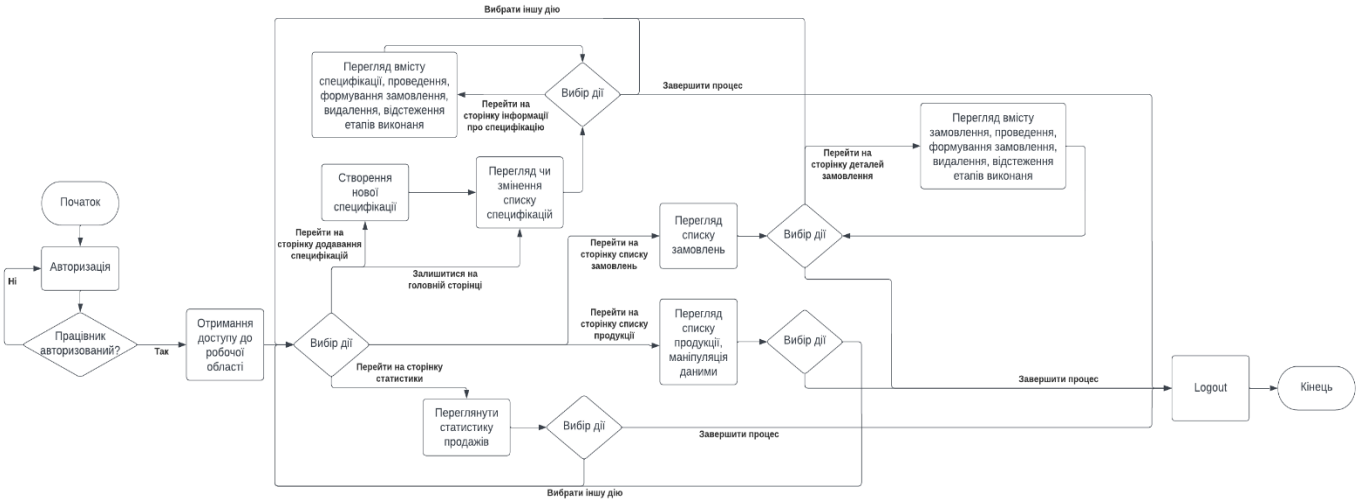

Рисунок 3.2 – Схема алгоритму послідовної поведінки менеджера із замовлень

Послідовність використання елементів веб-сервісу має наступні етапи:

– блок 1 Авторизація. Аналогічний блоку 1, описаному в п.п 3.2.2;

– блок 2 Перевірка на наявність користувача в системі. Аналогічний блоку 2, описаному в п.п 3.2.1;

– блок 3 Перегляд чи змінення списку специфікацій. Базова сторінка працівника. Всі сформовані специфікації клієнтських замовлень відображаються в

цьому списку. Є можливість видалення специфікації зі списку (але не з бази даних) та отримання детальної інформації щодо специфікації за кліком;

– блок 4 Створення нової специфікації. Припускається, що клієнт звертається до менеджера з замовлень через будь-який засіб зв'язку, не передбачений розроблюваною системою, та надає заяву на виготовлення продукції. Після цього менеджер може скористатися даною сторінкою, щоб занести отриману від замовника інформацію до бази;

– блок 5 Перегляд вмісту специфікації, проведення, формування замовлення, видалення, відстеження етапів виконання. На цій сторінці працівник може отримати детальну інформацію про сформовану специфікацію, включаючи етапи обробки та виготовлення, провести специфікацію та вручну сформувати та відправити pdf-документ із даними про замовлення або ж провести специфікацію та автоматично надіслати pdf-документ замовникові на електронну адресу. Також є можливим видалення специфікації з бази даних;

– блок 6 Перегляд списку замовлень. На цю сторінку потрапляють лише проведені специфікації, які відтепер ідентифікуються як замовлення. Є можливим перегляд існуючих замовлень та отримання детальної інформації за кліком;

– блок 7 Перегляд списку продукції, маніпуляція даними. На даній сторінці менеджер із замовлень може переглядати наявні послуги з виготовлення продукції, а також додавати нові, змінювати та видаляти поточні;

– блок 8 Перегляд статистики продажів. На цій сторінці можливо переглянути інформацію про прибуток за кожен місяць кожного року, сформовану, виходячи з виконаних замовлень, та представлену у вигляді лінійної діаграми;

– блок 9 Logout. Аналогічний блоку 12, описаному в п.п 3.2.1.

3.2.3 Розробка алгоритму поведінки планувальника виробництва

Для відображення послідовності дій планувальника виробництва при роботі з веб-сервісом розроблено алгоритм, який зображено на рисунку 3.3.

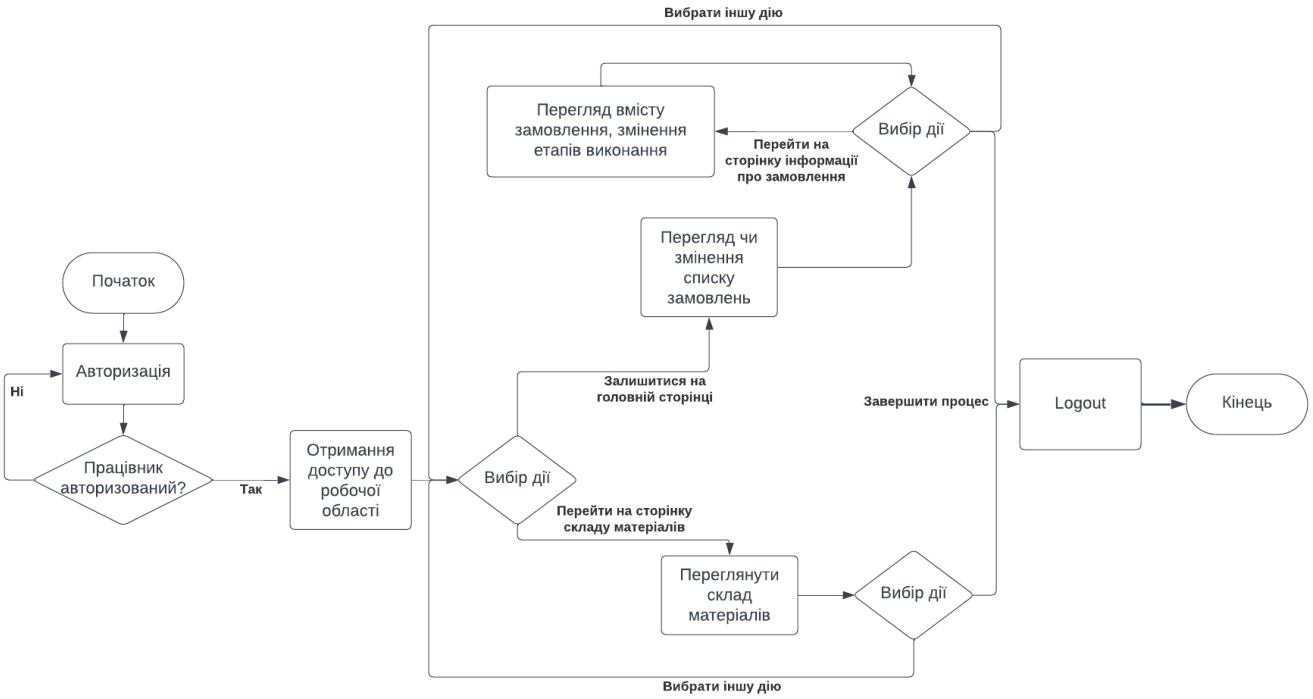

Рисунок 3.3 – Схема алгоритму послідовної поведінки планувальника виробництва

Послідовність використання елементів веб-сервісу має наступні етапи:

– блок 1 Авторизація. Аналогічний блоку 1, описаному в п.п 3.2.2;

– блок 2 Перевірка на наявність користувача в системі. Аналогічний блоку 2, описаному в п.п 3.2.1, і блоку 2, описаному в п.п. 3.2.2;

– блок 3 Перегляд чи змінення списку замовлень. Базова сторінка працівника. На ній планувальник виробництва бачить всі замовлення, які було допущено менеджером із замовлень до виконання. Є можливим видалення замовлення зі сторінки, але не з бази даних. Можливе отримання детальної інформації про замовлення за кліком;

– блок 4 Перегляд вмісту замовлення, змінення етапів виконання. На даній сторінці планувальник може побачити детальну інформацію про замовлення, а також керувати етапами виготовлення замовлення. Етапи, які було відмічено планувальником, відображаються на відповідній сторінці менеджера із замовлень.

Коли замовлення перейде в останній етап, робота планувальника над ним завершується, знов переходячи до менеджера з замовлень для фінальної обробки;

– блок 5 Переглянути склад матеріалів. В цілому, аналогічний блоку 3, описаному в п.п 3.2.1, однак за єдиним винятком – планувальник не має доступу до додавання, змінення та видалення матеріалів;

– блок 6 Logout. Аналогічний блоку 12, описаному в п.п 3.2.1, і блоку 9, описаному в п.п. 3.2.2.

3.3 Розробка логічної структури даних

Логічна структура даних – це концептуальне уявлення того, як дані організовані та структуровані в СУБД. Логічна структура даних є критично важливою, тому що вона визначає, як різні компоненти системи взаємодіятимуть між собою, забезпечуючи цілісність і ефективність обробки даних.

Структура даних в розроблюваній системі складається з шести колекцій: materials, orders, products, providers, specifications, users. Їх вигляд у хмарній СУБД MongoDB Atlas представлений на рисунку 3.4.

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
materials	2	184B	92B	36KB	1	36KB	36KB
orders	5	1.62KB	332B	36KB	1	36KB	36KB
products	2	212B	106B	36KB	1	36KB	36KB
providers	6	3.64KB	622B	36KB	1	36KB	36KB
specifications	16	5.11KB	328B	36KB	1	36KB	36KB
users	3	479B	160B	36KB	2	72KB	36KB

Рисунок 3.4 – Вигляд колекцій у СУБД MongoDB Atlas

Також дані колекції у вигляді документно-орієнтованої моделі даних представлено на рисунку 3.5.

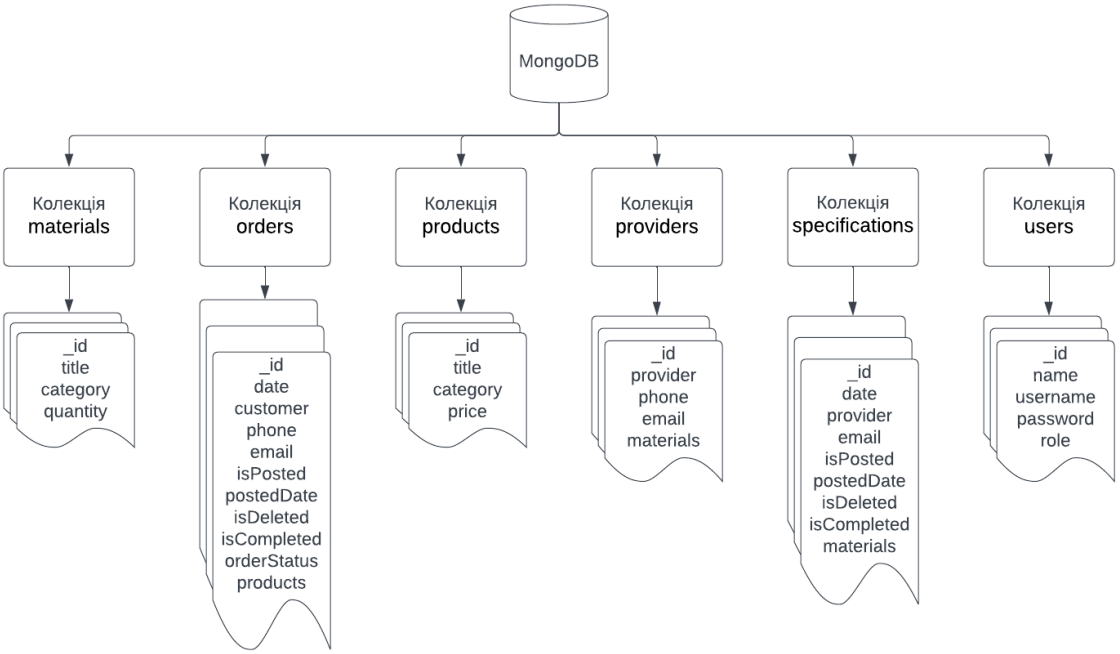

Рисунок 3.5 – Логічна структура даних

Колекція materials містить дані про матеріали, які компанія має у своєму розпорядженні. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює матеріал. Генерується автоматично при додаванні нових записів;

– title – назва матеріалу;

– category – категорія, до якої належить матеріал;

– quantity – кількість екземплярів даного матеріалу, які має у розпорядженні компанія.

Приклад екземпляру колекції materials зображено на рисунку 3.6.

```
_id: ObjectId('665799e540771595cec956f5')
title : "Гумовий ущільнювач"
category : "Гума"
quantity : 540
__v : 0
```

Рисунок 3.6 – Екземпляр колекції materials

Колекція orders містить інформацію про клієнтські замовлення, які приймає компанія. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює замовлення. Генерується автоматично при додаванні нових записів;

– date – бажана дата доставки, яку вказав клієнт, надаючи дані про замовлення. Компанія намагатиметься задовільнити дану потребу;

– customer – ПІБ замовника;

– phone – мобільний телефон замовника;

– email – адреса електронної пошти замовника;

– isPosted – маркер, який свідчить про початок обробки замовлення;

– postedDate – дата початку обробки замовлення;

– isDeleted – маркер для механізму soft delete, що застосовується у списках;

– isCompleted – маркер, що свідчить про завершення виконання замовлення;

– orderStatus – маркер, що свідчить про те, на якому етапі виконання перебуває замовлення;

– products – масив із продуктів, які клієнт побажав замовити.

Приклад екземпляру колекції orders зображено на рисунку 3.7.

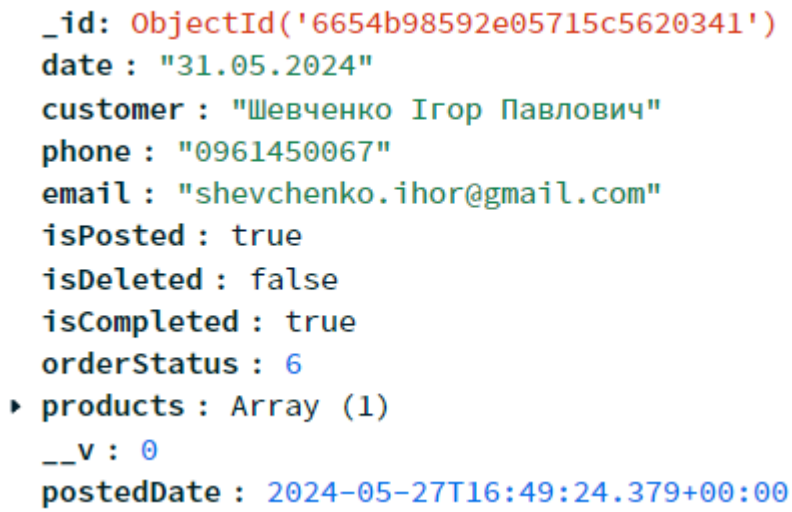

Рисунок 3.7 – Екземпляр колекції orders

Колекція products містить інформацію про продукцію, послугу з виготовлення якої надає компанія. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює продукт. Генерується автоматично при додаванні нових записів;

– title – назва продукту;

– category – категорія, до якої належить продукт;

– price – ціна за один примірник продукції.

Приклад екземпляру колекції products зображено на рисунку 3.8.

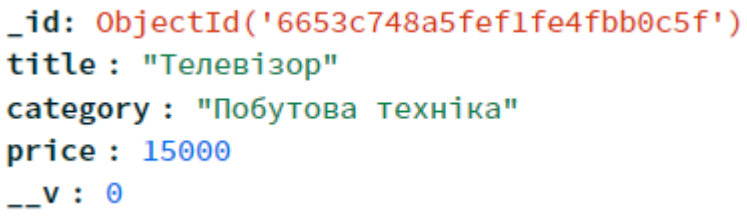

Рисунок 3.8 – Екземпляр колекції products

Колекція providers містить інформацію про постачальників, послугами яких користується компанія. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює постачальника. Генерується автоматично при додаванні нових записів;

– provider – назва компанії постачальника;

– phone – мобільний телефон постачальника;

– email – електронна пошта постачальника;

– materials – масив матеріалів, які можна буде замовити в постачальника.

Приклад екземпляру колекції products зображено на рисунку 3.9.

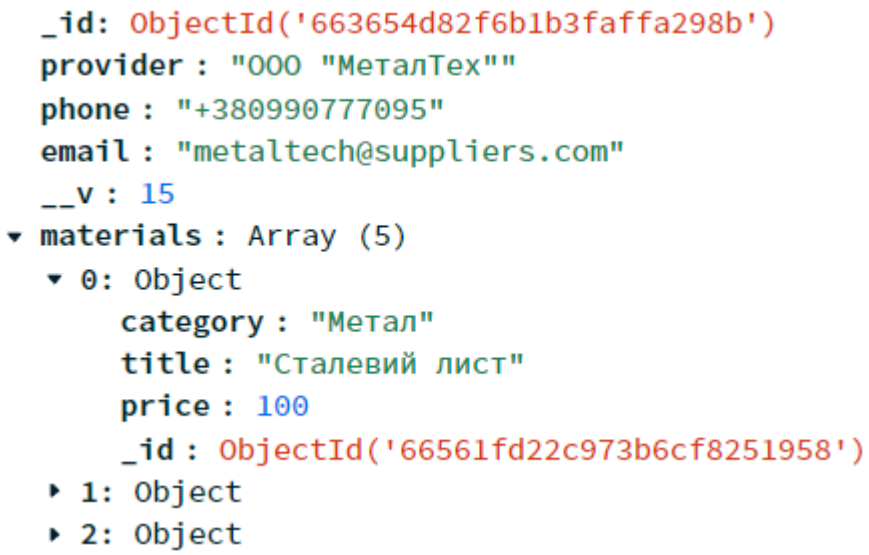

Рисунок 3.9 – Екземпляр колекції providers

Колекція specifications містить інформацію про специфікації та замовлення матеріалів компанією в постачальників. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює специфікацію. Генерується автоматично при додаванні нових записів;

– date – бажана дата доставки, яку вказав менеджер із закупівель, надаючи дані про замовлення. Компанія постачальника намагатиметься задовільнити дану потребу;

– provider – назва компанії постачальника;

– email – електронна адреса постачальника;

– isPosted – маркер, який свідчить про проведення специфікації;

– postedDate – дата проведення специфікації;

– isDeleted – маркер для механізму soft delete, що застосовується у списках;

– isCompleted – маркер, що свідчить про завершення виконання замовлення постачальником;

– materials – масив матеріалів, які менеджер із закупівель вказує при оформленні замовлення

Приклад екземпляру колекції specifications зображено на рисунку 3.10.

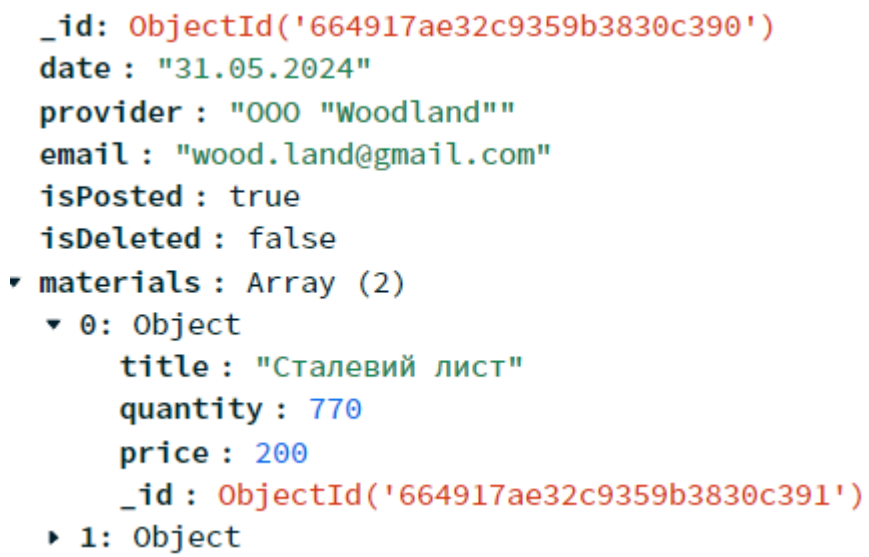

Рисунок 3.10 – Екземпляр колекції specifications

Колекція user містить особисті дані користувачів, які необхідні для авторизації в системі ERP. Детальний опис полів:

– _id – унікальний ідентифікатор, що ініціює специфікацію. Генерується автоматично при додаванні нових записів;

– name – ПІБ користувача;

– username – логін користувача;

– password – пароль користувача;

– role – маркер професії, за допомогою якого здійснюється перенаправлення користувача зі сторінки входу на робочий простір працівника.

Приклад екземпляру колекції users зображено на рисунку 3.11.

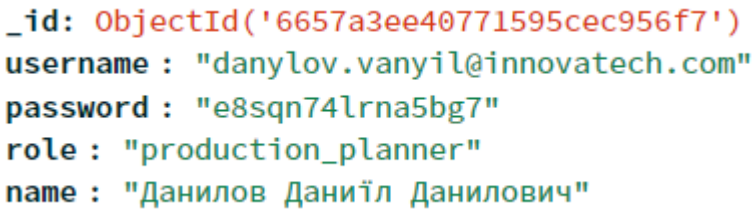

Рисунок 3.11 – Екземпляр колекції users

3.4 Визначення модулів, налаштування головного файлу app.js

Для ефективної реалізації проєкту бажаним є використання ряду модулів та фреймворків. Ці інструменти забезпечують базову функціональність додатку, включаючи обробку HTTP-запитів, управління сесіями, роботу з базою даних, обробку файлів та багато іншого. Усі необхідні модулі та фреймворки встановлюються та керуються за допомогою менеджера пакетів npm (Node Package Manager).

Npm – це онлайн-репозиторій для публікації відкритих Node.js-проектів та утиліта командного рядка для взаємодії зі згаданим репозиторієм, яка допомагає встановлювати пакунки та керувати версіями пакунків і залежностями [15]. У файлі package.json визначені всі залежності, необхідні для роботи проєкту, а також скрипти для запуску та розробки додатку (див. рисунок 3.12).

```
{
"name": "erp",
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
```

Рисунок 3.12 – Файл package.json

```
  "start": "node index",
"dev": "nodemon index"
 },
 "author": "",
 "license": "ISC",
 "dependencies": {
"csv-parser": "^3.0.0",
 "csvtojson": "^2.0.10",
  "express": "^4.18.2",
  "express-handlebars": "^7.1.2",
  "express-session": "^1.18.0",
  "mongoose": "^8.1.2",
  "multer": "^1.4.5-lts.1",
  "nodemailer": "^6.9.11",
  "passport": "^0.7.0",
 "passport-local": "^1.0.0",
 "pdfkit": "^0.14.0"
},
"devDependencies": {
"nodemon": "^3.0.3"
  }
}
```

Рисунок 3.12, аркуш 2

Детальний опис модулів, бібліотек та фреймворків:

– csv-parser – модуль для читання та обробки CSV-даних у Node.js;

– csvtojson – допомагає в перетворенні CSV у JSON;

– express – основний інструмент для створення веб-сервера та маршрутизації;

– express-handlebars – шаблонізатор для рендерингу динамічних HTML-сторінок на основі шаблонів;

– express-session – middleware для збереження інформації про сесії користувачів;

– mongoose – об'єктно-документний менеджер для MongoDB та Node.js;

– multer – middleware для обробки та збереження завантажених файлів на сервері;

– nodemailer – модуль для відправки електронних листів;

– passport – middleware для автентифікації із підтримкою різних стратегій;

– passport-local – стратегія локальної автентифікації;

– pdfkit – генератор pdf-документів;

– nodemon – інструмент для автоматичного перезавантаження Node.js-додатка при зміні файлів.

Вищезгадані модулі імпортуються в головному файлі проєкту – app.js, який включає в себе конфігурацію різноманітних компонентів, визначає маршрути для різних функціональних частин додатку, налаштовує та запускає сервер. Імпорт модулів наведено на рисунку 3.13.

```
const express = require('express')
const mongoose = require('mongoose')
const path = require('path')
const exphbs = require('express-handlebars')
const config = require('./config');
const session = require('express-session')
const passport = require('passport')
const LocalStrategy = require('passport-local').Strategy
```

Рисунок 3.13 – Імпорт модулів

Імпорт вищезгаданого middleware, допоміжних функцій, моделей та маршрутів проєкту наведено на рисунку 3.14.

```
const materialRoutes = require('./routes/materials')
const providerRoutes = require('./routes/providers')
const specificationRoutes = require('./routes/specifications')
const reportRoutes = require('./routes/reports')
const salesReportRoutes = require('./routes/sales-reports')
const orderRoutes = require('./routes/orders')
const prodOrderRoutes = require('./routes/prod-orders')
const productRoutes = require('./routes/products')
const authenticationRoutes = require('./routes/authentication')
const auth = require('./middleware/auth')
const { formatDate } = require('./service/formatDate')
const User = require('./models/User')
```

Рисунок 3.14 – Імпорт middleware, хелперів, моделей та маршрутів

Конфігурація порту, створення екземпляру Handlebars та його налаштування – базові операції при створенні проєкту. Їх наведено на рисунку 3.15.

```
const PORT = process.env.PORT || 8080
const app = express()
const hbs = exphbs.create({
 defaultLayout: 'main',
extname: 'hbs',
helpers: {
 formatDate,
 eq: function (a, b) {
return a === b;
 }
 }
})
app.engine('hbs', hbs.engine)
app.set('view engine', 'hbs')
app.set('views', 'views')
```

Рисунок 3.15 – Конфігурація порту, створення екземпляру Handlebars та його налаштування

В Express для обробки статичних файлів використовується функція express.static() (рисунок 3.16), яка вказує на каталог, що містить ці файли. Для статичних файлів було створено каталог public, у якому містяться файли для обробки стилів CSS, файли шрифтів, зображення та вивантажені файли.

```
app.use(express.urlencoded({ extended: true }))
app.use(express.static(path.join(__dirname, 'public')))
```

Рисунок 3.16 – Налаштування middleware для обробки даних і статики

За допомогою Passport.js було реалізовано автентифікацію. Налаштування сесій, автентифікації, а також стратегію Passport.js наведено на рисунку 3.17.

```
app.use(session({
 secret: config.session.secret,
resave: false,
 saveUninitialized: true
}));
app.use(passport.initialize())
app.use(passport.session())
passport.use(new LocalStrategy(
async (username, password, done) => {
 try {
  const user = await User.findOne({ username })
 if (!user) {
   return done(null, false, { message: `User ${username} not found` })
  }
     if (user.password !== password) {
    return done(null, false, { message: 'Incorrect password' })
  }
```

Рисунок 3.17 – Налаштування сесій, автентифікації, стратегії Passport.js

```
  return done(null, user)
   } catch (err) {
          return done(err)
}
   }
  ))
  passport.serializeUser((user, done) => {
    done(null, user.id)
  })
  passport.deserializeUser(async (id, done) => {
    try {
     const user = await User.findById(id)
    done(null, user)
   } catch (err) {
     done(err)
    }
  })
  app.use((req, res, next) => {
    res.locals.isAuthenticated = req.isAuthenticated();
    next();
  });
```

Рисунок 3.17, аркуш 2

Використання маршрутів та функцію запуску серверу наведено на рисунку 3.18.

```
app.use(authenticationRoutes,       materialRoutes,       providerRoutes,
specificationRoutes,    reportRoutes,    orderRoutes,    productRoutes,
prodOrderRoutes, salesReportRoutes)
async function start() {
 try {
   await mongoose.connect(config.mongodb.uri);
   app.listen(PORT, () => {
   console.log('Server has been started...');
```

Рисунок 3.18 – Використання маршрутів та функція запуску серверу

```
});
 } catch (e) {
   console.log(e);
  }
}
start()
```

Рисунок 3.18, аркуш 2

Ключі доступу до кластеру бази даних, а також секретний код для підпису сесій винесено в окремий файл config.js для забезпечення безпеки даних (рисунок 3.19).

```
module.exports = {
  mongodb: {
    uri: 'mongodb+srv://Username:Password@cluster0.ibqjpdg.mongodb.net/todos'
  },
  session: {
    secret: 'your-secret-key'
  }
}
```

Рисунок 3.19 – Файл конфігурації

3.5 Реалізація автентифікації та авторизації користувачів

Для забезпечення окремих рівнів доступу для користувачів кожної з трьох професій, було прийнято рішення впровадити механізм перевірки логіну, паролю, а також закріпленої за користувачем ролі. Спочатку система перевіряє відповідність внесених користувачем даних записам у базі даних, та, у разі позитивної відповіді, перенаправляє користувача зі сторінки входу на головну сторінку його робочої області в залежності від наявної ролі.

На стороні серверу автентифікацію та авторизацію забезпечують декілька файлів. Перший з них – контролер authController (рисунок 3.20), який відповідає за

обробку запиту автентифікації користувача та перенаправлення за відповідним маршрутом в залежності від ролі.

```
const User = require('../models/User');
const passport = require('passport');
class authController {
  async login(req, res, next) {
    passport.authenticate('local', (err, user, info) => {
      if (err || !user) {
        return res.render('login', { layout: 'login-layout', error: info ? info.message : 'Login failed' });
      }
      req.logIn(user, (err) => {
        if (err) {
          return next(err);
        }
        switch (user.role) {
          case 'order_manager':
            return res.redirect('/order-specifications');
          case 'production_planner':
            return res.redirect('/prod-orders-list');
          default:
            return res.redirect('/');
        }
      });
    })(req, res, next);
  }
  async getUsers(req, res) {
    try {
      const users = await User.find();
        res.json(users);
    } catch (e) {
      console.log(e)
    }
  }
}
module.exports = new authController()
```

Рисунок 3.20 – Контролер authController

Другим файлом є middleware-функція auth (рисунок 3.21), яка перевіряє, чи автентифікований користувач. Якщо користувач автентифікований – виконується відповідний обробник маршруту, якщо ж ні – користувач перенаправляється на сторінку авторизації.

```
module.exports = function(req, res, next) {
 if (req.isAuthenticated()) {
  return next();
 }
 res.redirect('/login');}
```

Рисунок 3.21 – Функція auth

Наступна middleware-функція – checkRole (рисунок 3.22). Її задача полягає в перевірці ролі користувача для надання доступу до конкретних сторінок. Це зроблено для того, щоб користувач, який вже отримав доступ до системи, не зміг перейти до ніяких інших робочих областей, окрім власної.

```
module.exports = function(allowedRoles) {
 return function(req, res, next) {
  if (req.isAuthenticated() && allowedRoles.includes(req.user.role)) {
   return next();
  }
  res.status(403).send('Access denied');
 }
}
```

Рисунок 3.22 – Функція checkRole

Збірником усіх маршрутів, пов’язаних з механізмом автентифікації та авторизації, є файл authentication, який зображено на рисунку 3.23.

```
const Router = require('express');
const router = new Router();
const controller = require('../controllers/authController');
router.get('/authentication', async (req, res) => { res.render('authentication');});
router.get('/login', (req, res) => {
 res.render('login', { layout: 'login-layout' });});
router.post('/login', controller.login)
router.get('/users', controller.getUsers)
router.get('/logout', (req, res) => {
 req.logout((err) => {
  if (err) { return next(err);}
  res.redirect('/login'); });});
module.exports = router
```

Рисунок 3.23 – Файл-роутер authentication

3.6 Розробка серверної частини проєкту

Бібліотека Mongoose є ODM-бібліотекою, що дає змогу розробляти моделі даних для роботи серверної частини проєкту з базою даних. Моделі створюються у вигляді об’єктів JavaScript та слугують своєрідними обгортками над схемами, які надають інтерфейс для взаємодії з колекцією документів. Створення схеми для визначення структури документів в колекціях здійснюється за допомогою інтерфейсу Schema, в якому прописуються поля та їх властивості. Приклад моделі даних, розроблений спеціально для втілення механізму автентифікації та авторизації, який було описано в п. 3.5, наведено на рисунку 3.24.

```
const {Schema, model} = require('mongoose');

const User = new Schema({
  name: {
    type: String,
    required: true
  },

  username: {
    type: String,
    unique: true,
    required: true
  },

  password: {
    type: String,
    required: true
  },

  role: {
    type: String,
    required: true
  }
})

module.exports = model('User', User);
```

Рисунок 3.24 – Модель даних User колекції users

Після компіляції моделі її можна використовувати з CRUD-операціями. Приклад застосування моделі User можна побачити у контролері authController на рисунку 3.20.

Взаємодія серверної частини проєкту із базою даних відбувається в декілька етапів:

– створення з’єднання з БД у головному файлі проєкту (рисунок 3.18) за допомогою ключів доступу (рисунок 3.19);

– отримання колекцій та їх об'єктів. Для цього зазвичай використовуються методи на кшталт find (повертає документи, що відповідають критеріям запиту), findById (повертає документ за його `_id`), findByIdAndUpdate (знаходить документ за `_id`та оновлює його), findByIdAndDelete (знаходить документ за `_id` та видаляє його).

– виконання CRUD-операцій;

– обробка помилок.

Приклад отримання об'єкту колекції з метою виконання CRUD-операції на оновлення за допомогою методу findByIdAndUpdate наведено на рисунку 3.25.

```
router.put('/provider/:id', async (req, res) => {
  try {
    const providerId = req.params.id;
    const { provider, phone, email } = req.body;
    const updatedProvider = await Provider.findByIdAndUpdate(
      providerId,
      { provider, phone, email },
      { new: true }
    );
    res.json({ success: true, provider: updatedProvider });
  } catch (err) {
    console.error(err);
    res.status(500).json({ success: false, msg: 'Сталася помилка під час оновлення
інформації про постачальника' });
  }
});
```

Рисунок 3.25 – Роутер з маршрутом PUT для оновлення інформації про постачальника

Приклад виведення списку однорідних документів за допомогою методу find зображено на рисунку 3.26.

```
router.get('/prod-orders-list', auth, checkRole(['production_planner']), async (req, res) => {
  const orders = await Order.find({ isDeleted: false, isPosted: true }).lean();
    res.render('prod-orders-list', {
      layout: 'prod-plnr.hbs',
      title: 'Замовлення',
      isOrders: true,
      orders,
      name: req.user.name
    });
});
```

Рисунок 3.26 – Роутер з маршрутом GET для отримання списку замовлень, що поступили на виготовлення

Роутер, що знаходить потрібний документ за ідентифікатором та видаляє його за допомогою методу findByIdAndDelete наведено на рисунку 3.27.

```
router.delete('/api/products/:id', auth, checkRole(['order_manager']), async (req, res) => {
  try {
    const { id } = req.params;
    const deletedProduct = await Product.findByIdAndDelete(id);
    if (!deletedProduct) {
      return res.status(404).json({ message: "Продукт не знайдено" });
    }
    res.status(200).json({ message: "Продукт успішно видалено" });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

Рисунок 3.27 – Роутер з маршрутом DELETE для отримання продукта та вилучення його з бази даних

Приклад застосування методу findById з метою створення нової специфікації на основі внесених користувачем даних, зображений на рисунку 3.28.

```
router.post('/order', async (req, res) => {
  try {
    const { delivery_date, provider, materials } = req.body;
    const providerData = await Provider.findById(provider);
    const materialsData = materials.map(item => ({
      title: item.title,
      quantity: parseInt(item.quantity),
      price: parseFloat(item.price)
    }));
    const specification = new Specification({
      date: delivery_date,
      provider: providerData.provider,
      email: providerData.email,
      isPosted: false,
      materials: materialsData
    });
    await specification.save();
  } catch (error) {
    console.error('Помилка під час створення специфікації:', error);
    res.status(500).json({ error: 'Помилка під час створення специфікації' });
  }
});
```

Рисунок 3.28 – Роутер з маршрутом POST для додавання нової специфікації в колекції specifications

Метод save, який показано на рисунку 3.28, використовується для збереження нового документу в базі даних.

В пункті 3.4 було зазначено, що серед залежностей проєкту значиться PDFkit, необхідний для генерації стилізованих звітів у форматі PDF. Для використання даної бібліотеки після її підключення необхідно створити файл-сервіс, задати в ньому необхідні стилі, забезпечити коректне передання даних, які мають записуватися в звіт. Приклад функції buildPDF, яка приймає дані специфікації замовлення компанії, викликає функції генерації заголовку, двох блоків інформації про замовлення та підвалу заяви зі вставленою в них інформацією зі специфікації, представлено на рисунку 3.29.

```
function buildPDF(specification, dataCallback, endCallback) {
  const doc = new PDFDocument({ deflateLevel: 9 });
  doc.on('data', dataCallback);
  doc.on('end', endCallback);
  doc.registerFont('DejaVuSans', './public/fonts/djsans/DejaVuSans.ttf');
  doc.registerFont('DejaVuSans-Bold', './public/fonts/djsans/DejaVuSans-
Bold.ttf');
  doc.registerFont('helvetica_cyr_oblique',
'./public/fonts/helvetica/helvetica_cyr_oblique.ttf');
  generateHeader(doc);
  generateOrderInformation(doc, specification);
  generateOrderTable(doc, specification);
  const currentHeight = doc.y;
  const footerHeight = 30;
  const maxHeight = doc.page.height - 50;
  if (currentHeight + footerHeight > maxHeight) {
    doc.addPage();
  }
  generateFooter(doc);
  doc.end();
}
```

Рисунок 3.29 – Функція buildPDF із файлу-сервісу pdf-service

Саме функція buildPDF викликатиметься в інших файлах проєкту для збирання готового звіту з інформацією про замовлення компанії. Спосіб її використання можна побачити на рисунку 3.30 у складі роутеру.

```
router.get('/specification/:id/pdf', auth, checkRole(['purchase_manager']), async
(req, res) => {
  try {
    const specificationId = req.params.id;
    const specification = await Specification.findById(specificationId);
    if (!specification) {
      return res.status(404).send('Специфікація не знайдена');}
    specification.isPosted = true;
    await specification.save();
    const pdfBuffer = await new Promise((resolve, reject) => {
      const chunks = [];
      const dataCallback = (chunk) => chunks.push(chunk);
      const endCallback = () => resolve(Buffer.concat(chunks));
      pdfService.buildPDF(specification, dataCallback, endCallback);
    });
    res.setHeader('Content-Type', 'application/pdf');
    res.setHeader('Content-Disposition', 'attachment;
filename="specification.pdf"');
    res.send(pdfBuffer);
  } catch (err) {
    console.error(err);
    res.status(500).send('Internal Server Error');
  }
});
```

Рисунок 3.31 – Роутер для отримання PDF-звіту з замовлення

Подібний механізм генерації pdf-звіту та його отримання використовується і в файлі-сервісі pdf-service-client. Його задача – формування звітів вже для клієнтських замовлень.

В пункті 3.4 також було згадано модуль Nodemailer у якості залежності проєкту. В розроблюваній системі його наявність необхідна для втілення можливості відправлення сформованих звітів на адресу електронної пошти постачальника або клієнта.

Перед використанням необхідно створити транспортер, обравши поштовий сервіс та акаунт, з адреси якого й буде здійснюватися відправлення повідомлень (рисунок 3.32).

```
const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'yaroslav.olinkevych@nure.ua,
    pass: 'hajx lyqr whoi fmri'
  }
});
```

Рисунок 3.32 – Створення транспортера

У тілі роутеру, зображеному на рисунку 3.33, здійснюється відправлення повідомлення, яке містить вкладений PDF-файл із деталями замовлення компанії на адресу постачальника, та текстовий коментар до замовлення, отриманий із форми користувача.

```
router.post('/send-specification/:id', async (req, res) => {
  try {
    const specificationId = req.params.id;
    const specification = await Specification.findById(specificationId).lean();
    if (!specification) {
      return res.status(404).json({ message: 'Специфікацію не знайдено' }); }
    const pdfBuffer = await new Promise((resolve, reject) => {
      const chunks = [];
      const dataCallback = (chunk) => chunks.push(chunk);
      const endCallback = () => resolve(Buffer.concat(chunks));
      pdfService.buildPDF(specification, dataCallback, endCallback);});
    const pdfUint8Array = new Uint8Array(pdfBuffer);
    const attachment = {
      filename: `specification_${specificationId}.pdf`,
      content: pdfUint8Array};
    const mailOptions = {
      from: 'yaroslav.olinkevych@nure.ua',
      to: specification.email,
      subject: 'Нова специфікація',
      text: 'Вкладений PDF-файл містить деталі замовлення.' + (req.body.comment ? `\n\nКоментар
до замовлення: ${req.body.comment}` : ''),
      attachments: [attachment] };
    await transporter.sendMail(mailOptions);
    res.status(200).json({ message: 'Специфікацію успішно надіслано на email' });
  } catch (error) {
    console.error('Помилка під час відправлення специфікації:', error);
    res.status(500).json({ error: 'Помилка під час відправлення специфікації' });}});
```

Рисунок 3.33 – Роутер для відправлення повідомлення постачальникові

За таким самим принципом створювалася решта роутерів, контролерів, моделей даних, допоміжних функцій та функцій-сервісів, які згодом зв’язувалися з клієнтською стороною, тим самим дозволяючи працівникам користуватися реалізованою серверною логікою прийнятним чином – за допомогою графічного інтерфейсу користувача.

3.7 Розробка клієнтської частини проєкту

Щоб надати користувачам ERP-системи можливість роботи з логікою, яку було створено на стороні серверу, необхідною є реалізація графічного інтерфейсу для кожної з робочих областей кожного представника професії. Для цього, за допомогою технології Handlebars для фреймворку Express. Файли Handlebars (.hbs)

в файловій системі проєкту на Express прийнято зберігати в теці views (рисунок 3.34).

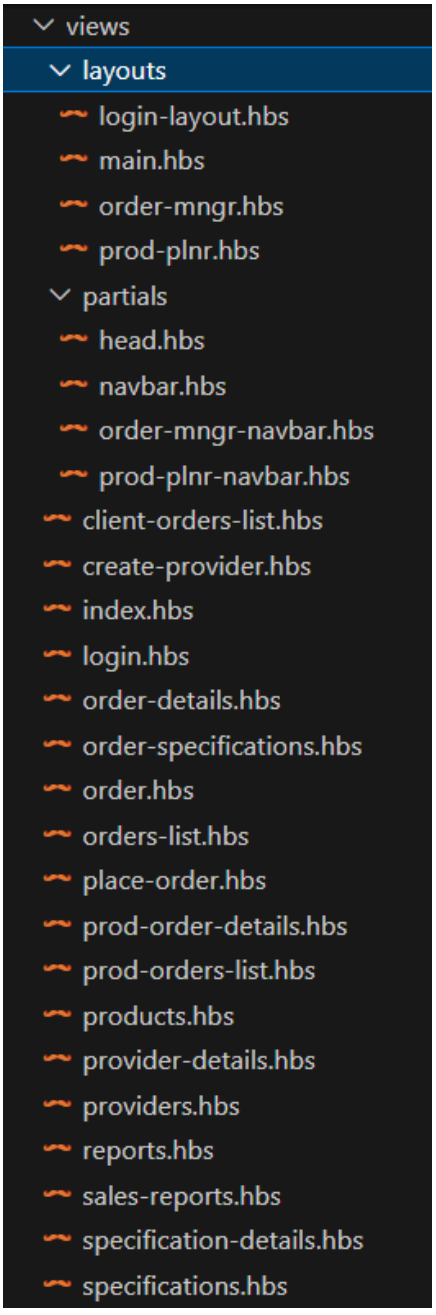

Рисунок 3.34 – Структура та вміст теки views

В корені views знаходяться лише шаблони представлення для сторінок, які безпосередньо виводитимуться користувачам за запитом, у підпапці layouts містяться шаблони загальних макетів сторінок, а в теці partials розташовані часткові шаблони для компонентів, які наодноразово використовуються у складі загальних макетів. Приклад коду макету сторінки менеджера з закупівель зображено на рисунку 3.35.

```
{{> head}}
<body>
  {{> navbar}}
  <div class="container">
    {{{ body }}}
  </div>
</body>
</html>
```

Рисунок 3.35 – Код файлу макету сторінки менеджера з закупівель

Вирази типу {{{ body }}} чи {{> navbar}} дозволяють викликати та рендерити часткові шаблони з основного шаблону. Даний макет викликає шаблон шапки сторінки, в якому міститься її конфігурація (тип кодування, мета-теги, залежності, назву вкладки), шаблон навігаційної панелі та, власне, вміст сторінки.

Приклад коду шаблону шапки представлений на рисунку 3.36.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.j
s"></script>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min
.css">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  <link rel="stylesheet" href="/index.css">
  <title>{{ title }}</title>
</head>
```

Рисунок 3.36 – Код файлу-шапки head.hbs

Приклад коду шаблону панелі навігації для інтерфейсу менеджера з замовлень зображено на рисунку 3.37.

```
<nav class="blue darken-4 z-depth-3">
  <div class="nav-wrapper">
    <a href="/" class="brand-logo left">
      <img src="/images/logo.png" alt="Logo" class="logo"> </a>
    <ul id="nav-mobile" class="right hide-on-med-and-down">
      <li class="user-info">
        <i class="material-icons left">account_circle</i>{{name}}, менеджер із закупівель
      </li>
      {{#if isIndex}}
      <li class="active"><a href="/">Склад</a></li>
      {{else}}
      <li><a href="/">Склад</a></li>
      {{/if}}
      {{#if isCreate}}
      <li class="active"><a class='dropdown-trigger' href="/providers"
          data-target='dropdown-providers'>Постачальники</a></li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-providers' class='dropdown-content'>
        <li><a href="/create-provider">Додати</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/providers">Постачальники</a></li>
      </ul>
      {{else if isProviders}}
      <li class="active"><a class='dropdown-trigger' href="/providers"
          data-target='dropdown-providers'>Постачальники</a></li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-providers' class='dropdown-content'>
        <li><a href="/create-provider">Додати</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/providers">Постачальники</a></li>
      </ul>
      {{else}}
      <li><a class='dropdown-trigger' href="/providers" data-target='dropdown-
providers'>Постачальники</a></li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-providers' class='dropdown-content'>
        <li><a href="/create-provider">Додати</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/providers">Постачальники</a></li>
      </ul>
      {{/if}}
      {{#if isOrder}}
      <li class="active"><a class='dropdown-trigger' href='/order' data-target='dropdown-
orders'>Замовлення</a></li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-orders' class='dropdown-content'>
        <li><a href="/order">Замовити</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/specifications">Специфікації</a></li>
      </ul>
      {{else if isSpecifications}}
      <li class="active"><a class='dropdown-trigger' href='/order' data-target='dropdown-
orders'>Замовлення</a></li>
```

Рисунок 3.37 – Код панелі навігації для менеджера з замовлень

```
 <ul id='dropdown-orders' class='dropdown-content'>
        <li><a href="/order">Замовити</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/specifications">Специфікації</a></li>
      </ul>
      {{else if isSpecificationDetails}}
      <li class="active"><a class='dropdown-trigger' href='/order' data-target='dropdown-
orders'>Замовлення</a></li>
      <ul id='dropdown-orders' class='dropdown-content'>
        <li><a href="/order">Замовити</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/specifications">Специфікації</a></li>
      </ul>
      {{else}}
      <li><a class='dropdown-trigger' href='/order' data-target='dropdown-
orders'>Замовлення</a></li>
      <ul id='dropdown-orders' class='dropdown-content'>
        <li><a href="/order">Замовити</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/specifications">Специфікації</a></li>
      </ul>
      {{/if}}
      {{#if isOrders}}
      <li class="active"><a class='dropdown-trigger' href='/orders-list' data-
target='dropdown-reports'>Статистика</a>
      </li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-reports' class='dropdown-content'>
        <li><a href="/orders-list">Закупівлі</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/reports">Звітність</a></li>
      </ul>
      {{else if isReports}}
      <li class="active"><a class='dropdown-trigger' href='/orders-list' data-
target='dropdown-reports'>Статистика</a>
      </li>
      <ul id='dropdown-reports' class='dropdown-content'>
        <li><a href="/orders-list">Закупівлі</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/reports">Звітність</a></li>
      </ul>
      {{else}}
      <li><a class='dropdown-trigger' href='/orders-list' data-target='dropdown-
reports'>Статистика</a>
      </li>
      <!-- Dropdown Structure -->
      <ul id='dropdown-reports' class='dropdown-content'>
        <li><a href="/orders-list">Закупівлі</a></li>
        <li class="divider" tabindex="-1"></li>
        <li><a href="/reports">Звітність</a></li>
      </ul>
      {{/if}}
```

Рисунок 3.37, аркуш 2

```
      {{#if isAuthenticated}}
      <li><a href="/logout">Вийти</a></li>
      {{/if}}
    </ul>
  </div>
</nav>
<script>
  document.addEventListener('DOMContentLoaded', function () {
    var elems = document.querySelectorAll('.dropdown-trigger');
    var instances = M.Dropdown.init(elems, {
      coverTrigger: false,
      hover: true,
      inDuration: 500});});
  document.addEventListener('DOMContentLoaded', function () {
    var elems = document.querySelector('nav');
    var instances = M.Pushpin.init(elems, {
      top: elems.offsetTop});});
</script>
```

Рисунок 3.37, аркуш 3

В даному коді прописані доступні користувачеві вкладки, маршрути для перенаправлення за кліком, умови активності для виділення поточної вкладки іншим кольором, отримання та виведення з бази даних ПІБ працівника, посилання на логотип, пушпін-функція, що дозволяє закріпити панель навігації на сторінці під час скролінгу, а також функція для використання випадних списків.

Приклад коду body, який є основним інформаційним блоком, що представляє сторінку списку замовлень, які компанія зробила в постачальників, наведено на рисунку 3.38.

```
      <h2>Список заявок</h2>
{{#if specifications.length}}
<table class="orders">
  <tr class="orders-header">
    <td>
      <div class="input-field">
        <input placeholder="Введіть постачальника" id="provider_search"
type="text" class="validate">
        <label for="provider_search" style="display: flex; align-items:
center;">Постачальник <i
            class="material-icons">search</i></label>
      </div>
    </td>
    <td>
      <div class="input-field">
        <input type="text" id="date" name="postedDate" class="datepicker"
autocomplete="off">
```

Рисунок 3.38 – Код сторінки списку замовлень компанії

```
      <label>Дата подання заявки</label>
    </div>
  </td>
  <td>
    <div class="input-field">
      <select id="deliveryDateSort">
        <option value="all" selected>Всі</option>
        <option value="newer">Поточні</option>
        <option value="older">Прострочені</option>
      </select>
      <label>Бажана дата доставки</label>
    </div>
  </td>
  <td>
    <div class="input-field">
      <select id="completedSort">
        <option value="all" selected>Всі</option>
        <option value="completed">Виконано</option>
        <option value="notCompleted">Не виконано</option>
      </select>
      <label>Статус</label>
    </div>
  </td>
</tr>
{{#each specifications}}
<tr data-id="{{_id}}" class="orderRow" data-provider="{{provider}}" data-
posteddate="{{formatDate postedDate}}">
    <td>{{provider}}</td>
    <td>{{formatDate postedDate}}</td>
    <td>{{date}}</td>
    <td>{{#if isCompleted}}Виконано{{else}}Не виконано{{/if}}</td>
  </tr>
  {{/each}}
</table>
{{else}}
<p>У базі не зареєстровано жодного замовлення.</p>
{{/if}}
<script>
 document.addEventListener('DOMContentLoaded', function () {
  var elems = document.querySelectorAll('select');
  var instances = M.FormSelect.init(elems);
  var datepicker = document.querySelectorAll('.datepicker');
  M.Datepicker.init(datepicker, {
    format: 'dd.mm.yyyy',
    showClearBtn: true,
    i18n: {
      cancel: 'Відміна',
      clear: 'Очистити',
      done: 'Готово',
      months: ["Січень", "Лютий", "Березень", "Квітень", "Травень", "Червень",
```

Рисунок 3.38, аркуш 2

```
"Липень", "Серпень", "Вересень", "Жовтень", "Листопад", "Грудень"],
      monthsShort: ["Січ", "Лют", "Бер", "Квт", "Трав", "Чера", "Лип", "Сер",
"Вер", "Жов", "Лис", "Груд"],
      weekdays: ["Неділя", "Понеділок", "Вівторок", "Середа", "Четвер",
"П'ятниця", "Субота"],
      weekdaysShort: ["Нд", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"],
      weekdaysAbbrev: ["Н", "П", "В", "С", "Ч", "П", "С"]}});
  const providerSearchInput = document.getElementById('provider_search');
  const datepickerInput = document.getElementById('date');
  const deliveryDateSort = document.getElementById('deliveryDateSort');
  const completedSort = document.getElementById('completedSort');
  const orderRows = document.querySelectorAll('.orderRow');
  function applyFilters() {
    const selectedProvider = providerSearchInput.value.toLowerCase();
    const selectedDate = datepickerInput.value;
    const selectedDeliveryDateSort = deliveryDateSort.value;
    const selectedCompletedSort = completedSort.value;
    orderRows.forEach(row => {
      const provider = row.dataset.provider.toLowerCase();
      const postedDate = row.dataset.posteddate;
      const deliveryDate = new
Date(row.children[2].textContent.trim().split('.').reverse().join('-'));
      const isCompleted = row.lastElementChild.textContent.trim() === 'Так';

      const currentDate = new Date();
      let shouldDisplay = true;
      if (selectedProvider && !provider.includes(selectedProvider)) {
        shouldDisplay = false;}
      if (selectedDate && postedDate !== selectedDate) {
        shouldDisplay = false;}
      if (selectedDeliveryDateSort === 'newer' && deliveryDate < currentDate) {
        shouldDisplay = false;
      } else if (selectedDeliveryDateSort === 'older' && deliveryDate >=
currentDate) {
        shouldDisplay = false;}
      if (selectedCompletedSort === 'completed' && !isCompleted) {
        shouldDisplay = false;
      } else if (selectedCompletedSort === 'notCompleted' && isCompleted) {
        shouldDisplay = false;}
      if (shouldDisplay) {
        row.style.display = '';
      } else {
        row.style.display = 'none';}});}
  providerSearchInput.addEventListener('input', applyFilters);
  datepickerInput.addEventListener('change', applyFilters);
  deliveryDateSort.addEventListener('change', applyFilters);
  completedSort.addEventListener('change', applyFilters);
  orderRows.forEach(row => {
    row.addEventListener('click', () => {
      const specificationId = row.dataset.id;
      window.location.href = `/specification/${specificationId}`});});});
</script>
```

Рисунок 3.38, аркуш 3

Даний код реалізує відображення таблиці списку замовлень, які викликаються з бази даних, перед цим пройшовши перевірку на відповідність до умови проведеності. Завдяки застосованим механізмам фільтрації, кожну колонку таблиці можливо відсортувати за ім’ям постачальника, датою подання заявки, бажаною датою доставки та статусом. За кліком по конкретному рядку, можна отримати детальну інформацію про замовлення.

Для надання інтерфейсу прийнятного та зрозумілого вигляду було використано файли шрифтів, зображень та файл стилів, які в контексті додатку Express прийнято розміщувати в теці public (рисунок 3.39).

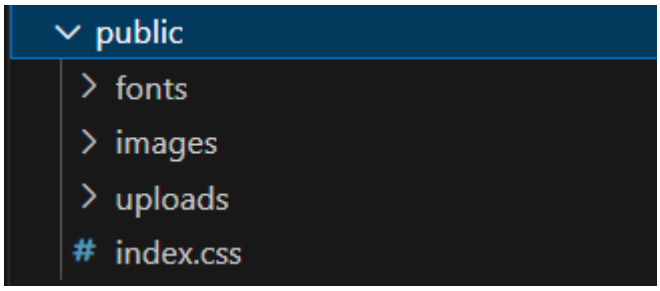

Рисунок 3.39 – Вміст теки public

Стилі для оформлення .hbs-файлів задаються за вказаними hmtl-тегами та їх класами. Фрагмент коду файлу index.css наведено на рисунку 3.40.

```
nav {
  position: fixed;
  top: 0;
  width: 100%;
  z-index: 1000;
}

.user-info {
  margin-right: 30px;
}
```

Рисунок 3.40 – Фрагмент коду файлу стилів index.css

Таким чином, якщо менеджер із закупівель захоче потрапити на сторінку зі списком замовлень компанії, йому потрібно буде здійснити наступні дії:

– пройти процедуру авторизації та отримати доступ до робочої області. Вигляд сторінки зображений на рисунку 3.41;

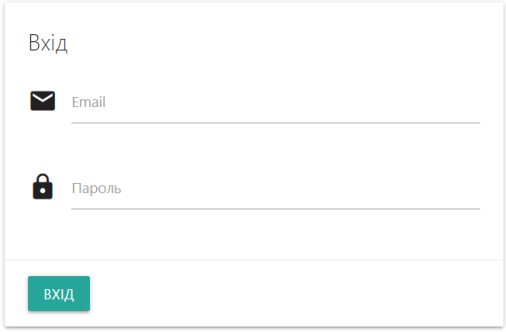

Рисунок 3.41 – Вигляд сторінки авторизації

– за допомогою панелі навігацію знайти у випадному списку сторінку списку замовлень та перейти на неї. Вигляд сторінки та принцип роботи випадних списків зображені на рисунку 3.42.

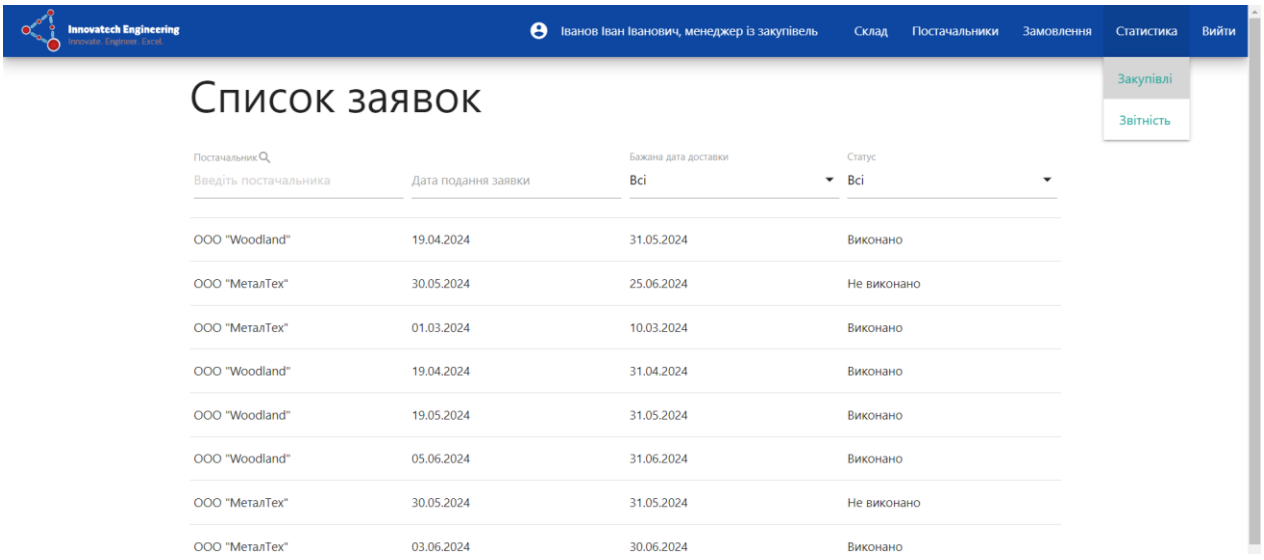

Рисунок 3.42 – Вигляд робочої області менеджера з закупівель із відкритою сторінкою списку замовлень компанії

Таким чином, опинившись на потрібній сторінці, працівник може виконати вищеописані дії, які було реалізовано за допомогою блоку функції у .hbs-файлі та відповідного роутера (рисунки 3.43 – 3.46).

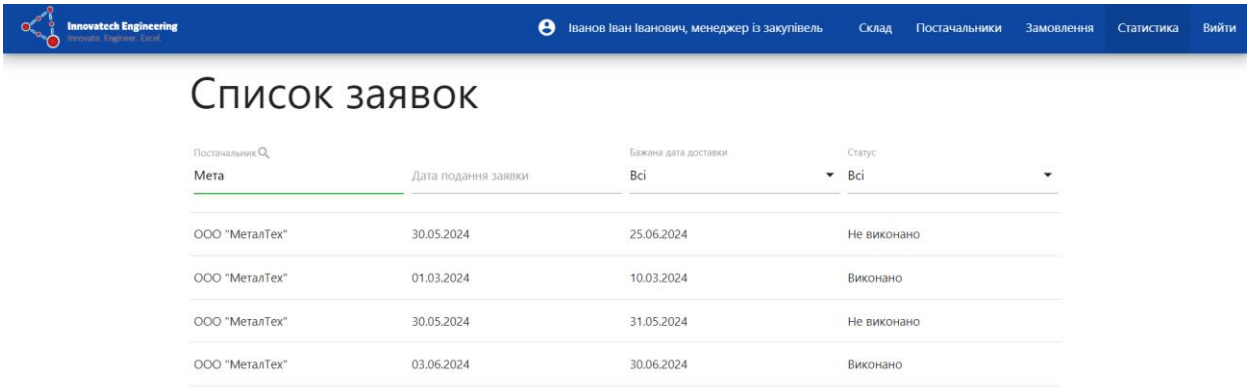

Рисунок 3.43 – Пошук заявок за компанією-постачальником

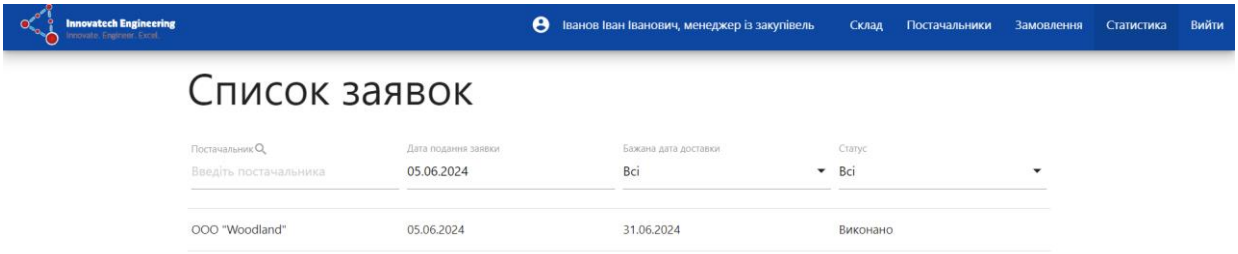

Рисунок 3.44 – Фільтрація замовлень за датою подання заявки

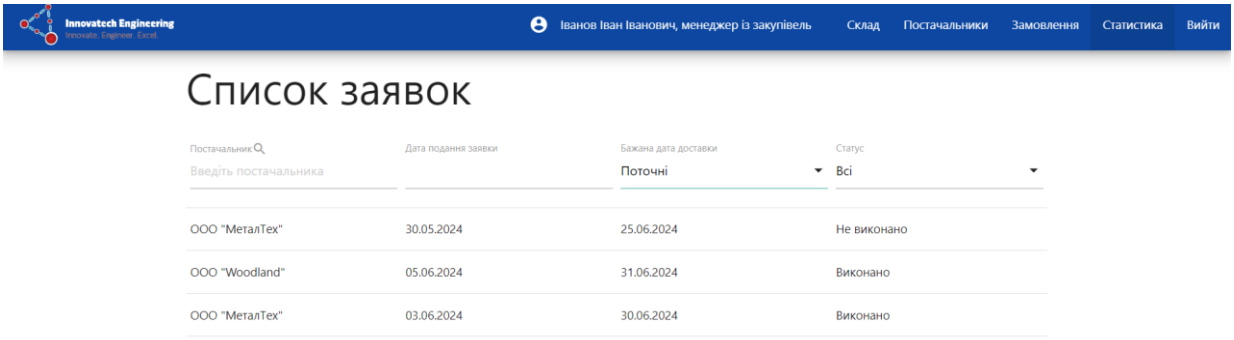

Рисунок 3.45 – Фільтрація замовлень за бажаною датою доставки

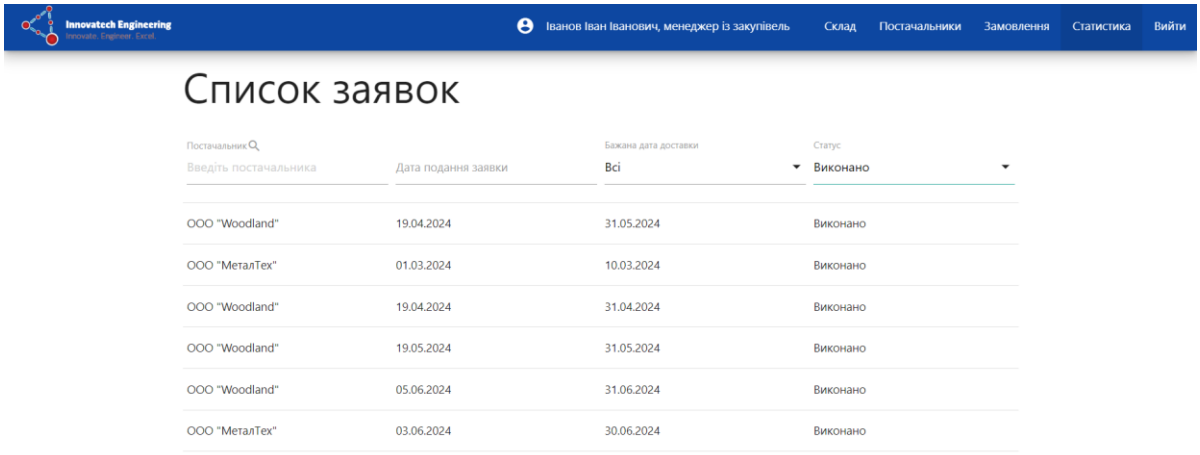

Рисунок 3.46 – Фільтрація замовлень за статусом

За таким самим принципом створювалася та пов’язувалася решта .hbs-файлів для реалізації інших сторінок робочої області менеджера з закупівель, а також сторінок для інтерфейсів менеджеру з замовлень та планувальника виробництва й самих інтерфейсів відповідно.

ВИСНОВКИ

Мета кваліфікаційної роботи полягала в розробці ERP-рішення на базі браузерного додатку, призначенням якого є забезпечення зручного середовища для моніторингу та управління ресурсами промислової компанії.

У ході виконання роботи було проведено ґрунтовний аналіз предметної області, дослідження існуючих ERP-систем, а також визначено основні вимоги до розроблюваної ERP-системи для оптимізації управління ресурсами промислового підприємства. Було розроблено функціональні вимоги до компонентів системи, зокрема для менеджера із закупівель, замовлень та планувальника виробництва. Використовуючи мову UML, було змодельовано роботу системи за допомогою діаграм прецедентів, послідовності, комунікації та класів. Це дозволило візуалізувати структуру системи, взаємодію між її компонентами та зв'язки між класами.

Було обґрунтовано вибір програмного стеку для розробки веб-сервісу, розроблені детальні алгоритми послідовної поведінки різних типів користувачів системи: менеджера із закупівель, менеджера із замовлень та планувальника виробництва. Ці алгоритми візуалізовані за допомогою схем, що чітко ілюструють взаємодію користувачів з різними компонентами веб-сервісу. Також була спроектована логічна структура даних у вигляді шести колекцій у документно-орієнтованій базі даних MongoDB. Для кожної колекції наведено опис полів та приклади документів, що забезпечує цілісність та ефективність обробки даних у системі.

Були визначені необхідні модулі, бібліотеки та фреймворки Node.js, налаштовано головний файл додатку app.js, включаючи конфігурацію порту, налаштування Handlebars, обробку статичних файлів, налаштування middleware для обробки даних, сесій та автентифікації за допомогою Passport.js, а також описано використання маршрутів та запуск серверу.

Після завершення етапів планування та проєктування було розроблено серверну та клієнтську частини проєкту, які, взаємодіючи, утворюють повністю функціональний вебсервіс, що цілком відповідає висунутим вимогам та, завдяки зручному графічному інтерфейсу та потужному функціоналу, здатний задовільнити потреби виробництва в оптимізації робочих процесів типу моніторингу ресурсів, планування та обробки замовлень, взаємодії з партнерами, отриманні звітності.

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 проектування 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. Олінкевич Я.В., Колесник Л.В. Інтеграція MongoDB та Node.js: сучасний підхід до розробки веб-додатків для оптимізації управління ресурсами промислової компанії // «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві», 22 листопада 2023. Харків, Україна. 2023. С. 202-223.

5. Олінкевич Я.В. Впровадження ERP-системи на виробництві // 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. Основи UML. KDE Documentation. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (дата звернення: 08.05.2024).

7. What is Unified Modeling Language. Lucidchart. URL: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language (дата звернення: 23.05.2024).

8. Maxym Z. Діаграма послідовності (Sequence Diagrams). Maxym Zosym. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 23.05.2024).

9. Діаграми UML для моделювання процесів і архітектури проекту. Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 24.05.2024).

10. What is javascript: A beginner's guide to the basics of JS. Hostinger Tutorials. URL:https://www.hostinger.com/tutorials/what-is-javascript (дата звернення 25.05.24).

11. What is node.js? - training. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/training/modules/intro-to-nodejs/2-what (дата звернення: 26.05.2024).

12. Express/Node introduction - Learn web development | MDN. *MDN Web Docs*. URL: http://surl.li/uaxve (дата звернення: 26.05.2024).

13. What Is MongoDB? All About the Popular Open Source Database. *Kinsta*®. URL: http://surl.li/uaxvj (дата звернення: 26.05.2024).

14. W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/mongodb/ (дата звернення: 27.05.2024).

15. What is NPM? The Complete Beginner's Guide. CareerFoundry. URL: https://careerfoundry.com/en/blog/web-development/what-is-npm/#what-is-npm (дата звернення: 27.05.2024).

16. Основи охорони праці: Підручник./ К.Н. Ткачук, М.О. Халімовський, В.В. Зацарний, Д.В. Зеркалов, Р.В. Сабарно, О.І. Полукаров, В.С. Коз‘яков, Л.О. Митюк; За ред. К.Н. Ткачука і М.О. Халімовського. – К.: Основа, 2003 – 180- 285 с.