НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Інформаційних систем і технологій

_____ Швиденко М.З., к. ек. наук, доцент, заслужений професор

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

На тему: «Веб система управління замовленнями для підприємства з виробництва металу»

Спеціальність 126 «Інформаційні системи та технології»

_____

Гарант освітньої програми

к. ек. наук, доцент / Мокрієв М.В /

(підпис) (ПІБ)

Керівник дипломного проекту: / Назаренко В.А /

(підпис) (ПІБ)

Виконав: / Іванина Д.О /

(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Завідувач кафедри інформаційних систем і технологій

_____ __Швиденко М.З.___
підпис ініціали та прізвище

__ _____ 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студенту(ці) _____Іванина Д.О._____

Спеціальності 126 "Інформаційні системи та технології"

1. Тема роботи: **«Веб система управління замовленнями для підприємства з виробництва металу»**

Затверджена наказом ректора від 16.12.2024р. №2245С

2. Термін подання завершеної роботи на кафедру – ___06.2025р.

3. Вихідні дані Розробити Веб систему управління замовленнями для підприємства з виробництва металу

4. Перелік питань, що розглядаються:

1. Теоретико-методологічні засади дослідження систем управління виробничими замовленнями

2. Архітектурно-технологічні аспекти проєктування веб-системи управління замовленнями

3. Програмна реалізація веб-системи управління замовленнями

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми, планування та підготовка	25.02.2025	Завдання виконано
2	Дослідження та збір матеріалів	30.03.2025	Завдання виконано
3	Аналіз та написання	20.05.2025	Завдання виконано
4	Попередній перегляд та оцінка	27.05.2025	Завдання виконано
5	Захист бакалаврської роботи	06.2025	Завдання виконано

Керівник кваліфікаційної роботи _____ / __Назаренко В.А., доктор філософії____
підпис ПІБ, вчене звання та ступінь

Завдання прийняв до виконання _____ / _____Іванина Д.О._____
підпис

Дата отримання завдання 15.02.2025

РЕФЕРАТ

Тема бакалаврської кваліфікаційної роботи: «Веб система управління замовленнями для підприємства з виробництва металу».

Автор роботи: Іванина Д.О.

Керівник роботи: Назаренко В.А.

Пояснювальна записка: 98 с., 25 рис., 5 табл., 21 джерело.

Графічна частина: 15 презентаційних слайдів.

Веб система управління замовленнями для підприємства з виробництва металу

У бакалаврській роботі досліджено проблему автоматизації управління виробничими замовленнями на підприємствах металообробної галузі. Запропоновано розробку веб-системи, що забезпечує централізовану обробку замовлень, інтеграцію функціональних підрозділів підприємства, підвищення точності техніко-економічних розрахунків та мінімізацію операційних витрат. У роботі представлено теоретико-методологічний аналіз предметної області, архітектурне проектування системи та програмну реалізацію із застосуванням сучасних технологій — Angular, Spring Boot та MongoDB. Розроблена система включає мультирольовий інтерфейс, REST API, автоматизовані модулі технічного й фінансового прорахунку, а також інструменти для аналітики та звітності. Отримані результати мають практичне значення для підвищення ефективності управління замовленнями в умовах сучасного виробництва.

Ключові слова: управління замовленнями, веб-система, автоматизація, Angular, Spring Boot, MongoDB, металообробка, бізнес-процеси.

This bachelor's thesis addresses the problem of automating the management of production orders at metalworking enterprises. A web-based system is proposed

to centralize order processing, integrate functional departments, improve the accuracy of technical and economic calculations, and minimize operational costs. The study includes a theoretical and methodological analysis of the subject area, system architecture design, and software implementation using modern technologies such as Angular, Spring Boot, and MongoDB. The developed system features a multi-role user interface, REST API, automated modules for technical and financial calculations, and tools for analytics and reporting. The results obtained have practical significance for enhancing order management efficiency in modern manufacturing environments.

Keywords: order management, web system, automation, Angular, Spring Boot, MongoDB, metalworking, business processes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ERP - *Enterprise Resource Planning* - система планування ресурсів підприємства.

CRM - *Customer Relationship Management* - система управління взаємовідносинами з клієнтами.

UI - *User Interface* - інтерфейс користувача.

UX - *User Experience* - користувацький досвід.

SPA - *Single Page Application* - односторінковий вебзастосунок.

API - *Application Programming Interface* - програмний інтерфейс прикладного програмування.

REST - *Representational State Transfer* - архітектурний стиль взаємодії клієнт-сервер.

MongoDB - документно-орієнтована база даних NoSQL.

HTTP - *HyperText Transfer Protocol* - протокол передавання гіпертексту.

JWT - *JSON Web Token* - формат токену для автентифікації користувача.

JSON - *JavaScript Object Notation* - формат представлення даних.

Angular - фреймворк для створення SPA-застосунків на JavaScript/TypeScript.

Spring Boot - фреймворк Java для створення бекенд-застосунків.

HTML/CSS - мови розмітки та стилізації вебсторінок.

JS/TS - JavaScript / TypeScript - мови програмування для фронтенду.

ВСТУП

Сучасні тенденції розвитку виробничого сектору економіки характеризуються посиленням конкурентної боротьби, глобалізацією ринків та підвищенням вимог споживачів до якості продукції та оперативності виконання замовлень. В умовах динамічних змін ринкового середовища металообробні підприємства потребують ефективних інструментів управління бізнес-процесами, зокрема, систем управління виробничими замовленнями, що забезпечують оптимізацію виробничого циклу, підвищення продуктивності праці та мінімізацію операційних витрат.

Аналіз існуючих практик управління замовленнями на металообробних підприємствах демонструє превалювання застарілих підходів, базованих на використанні електронних таблиць, паперової документації та локалізованих програмних рішень без функціональних можливостей синхронізації. Такий стан справ призводить до множини негативних наслідків: дублювання інформаційних потоків, втрати даних, ускладнення аналітичних процедур, надмірної залежності від людського фактору. Виникнення подібних дисфункцій на етапах калькуляції матеріальних витрат або прогнозування тривалості виробничого циклу спричиняє затримки у виконанні замовлень, перевитрати ресурсів та погіршення іміджевих характеристик підприємства.

Існуючі програмні рішення для управління виробничими замовленнями, такі як ProShop ERP, MRPeasy та Microsoft Dynamics 365, характеризуються або надмірним функціоналом, або недостатньою адаптивністю до специфічних особливостей бізнес-процесів конкретних підприємств. Це створює передумови для розробки спеціалізованих систем, здатних забезпечити оптимальний баланс між функціональною повнотою, простотою використання та економічною доцільністю впровадження.

Метою даної магістерської роботи є розробка веб-системи управління замовленнями для підприємств з виробництва металу, що забезпечить комплексну автоматизацію процесів прийому, обробки, планування та виконання замовлень. Система орієнтована на підвищення ефективності

операційної діяльності підприємств, оптимізацію використання ресурсів та підвищення рівня задоволеності клієнтів.

Об'єктом дослідження виступають процеси управління виробничими замовленнями на підприємствах металообробної галузі. Предметом дослідження є інформаційні технології та програмні засоби автоматизації процесів управління замовленнями.

Наукова новизна роботи полягає у розробці архітектурного рішення веб-системи управління замовленнями, адаптованого до специфіки металообробних підприємств, з урахуванням особливостей взаємодії функціональних підрозділів та процесів прийняття рішень. Практична значимість роботи визначається можливістю безпосередньої імплементації розробленої системи на підприємствах металообробної галузі з метою оптимізації бізнес-процесів та підвищення конкурентоспроможності.

Структурно робота складається з трьох розділів, що охоплюють теоретико-методологічні засади дослідження систем управління виробничими замовленнями, архітектурно-технологічні аспекти проєктування веб-системи та програмну реалізацію розробленого рішення. Перший розділ присвячено аналізу предметної області, існуючих інформаційних технологій та формуванню вимог до інструментальних засобів. У другому розділі висвітлено загальну концепцію системи, архітектурні рішення та процеси взаємодії компонентів. Третій розділ містить опис програмної реалізації, включаючи особливості імплементації функціональності, структуру коду та результати тестування системи.

Методологічною основою дослідження виступають загальнонаукові методи аналізу та синтезу, системний підхід до вивчення об'єкта дослідження, методи об'єктно-орієнтованого проєктування та патерни розробки програмного забезпечення. Інформаційною базою дослідження слугували наукові публікації з питань управління виробничими процесами, документація сучасних технологічних фреймворків та практичний досвід розробки інформаційних систем.

Робота викладена на 96 сторінках, містить 25 рисунків і 5 таблиць і складається зі вступу, трьох розділів, висновків та списку літератури з 21 найменування.

РОЗДІЛ 1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ ДОСЛІДЖЕННЯ СИСТЕМ УПРАВЛІННЯ ВИРОБНИЧИМИ ЗАМОВЛЕННЯМИ

1.1 Змістовний аналіз предметної області управління виробничими замовленнями

Управління виробничими замовленнями становить фундаментальний компонент операційної діяльності підприємств виробничого сектору, незалежно від галузевої приналежності та масштабів функціонування. Системне впорядкування та оптимізація процесів, пов'язаних із надходженням, обробкою, плануванням та реалізацією замовлень, суттєво детермінують ефективність виробничого циклу, конкурентоспроможність підприємства та його ринкову позицію. Аналіз сучасних тенденцій розвитку виробничого сектору демонструє, що в умовах турбулентності ринкового середовища та високої динаміки споживчих преференцій виробники повинні демонструвати підвищену адаптивність у контексті модифікації пропозиції, забезпечення належних якісних характеристик продукції, дотримання контрактних зобов'язань щодо термінів виконання та оптимізації використання внутрішніх ресурсів. Зазначені виклики актуалізують проблематику формування структурованої системи управління виробничими замовленнями, спроможної забезпечити комплексний контроль життєвого циклу виробу від моменту ініціації клієнтського запиту до етапу відвантаження готової продукції [1].

Практична імплементація концепції ефективного управління виробничими замовленнями супроводжується низкою об'єктивних перешкод та складнощів. Значна кількість суб'єктів господарювання виробничого сектору продовжує використовувати застарілі або фрагментовані інструменти управлінського впливу, серед яких: електронні таблиці Excel, документація на паперових носіях, локалізовані програмні рішення без функціональних можливостей синхронізації. Зазначений підхід генерує множину негативних

наслідків, зокрема: дублювання інформаційних потоків, втрату даних, ускладнення аналітичних процедур, надмірну залежність від людського фактору. Виникнення подібних дисфункцій на етапах калькуляції матеріальних витрат або прогнозування тривалості виробничого циклу може спричинити затримки у виконанні замовлень, перевитрати ресурсів, погіршення іміджевих характеристик підприємства та, як наслідок, втрату клієнтської бази [2].

Структурно-функціональна декомпозиція предметної області дозволяє виокремити наступні ключові компоненти: система прийому замовлень (акумулювання інформації щодо клієнта, формування специфікації), техніко-технологічний аналіз (калькуляція ресурсних потреб, селекція технологічного інструментарію), фінансово-економічний аналіз (вартісна оцінка замовлення), планування виробничого процесу (розподіл виробничих потужностей), система звітності (функції контролю та аналітичного забезпечення). Процесуальна складова управління замовленнями характеризується послідовністю взаємопов'язаних етапів, реалізація яких забезпечується різними функціональними підрозділами та посадовими особами, зокрема: менеджерами з продажу, інженерно-технічним персоналом, бухгалтерсько-економічним блоком, технологами, виробничим персоналом. Кожен структурний елемент системи характеризується власною функціональною специфікою, проте досягнення інтегрального результату детермінується рівнем узгодженості та координації процесів. Відповідно, системний підхід та автоматизація управлінських процедур набувають статусу ключових детермінант ефективності сучасного виробничого менеджменту [3].

Необхідність підвищення продуктивності праці, оптимізації операційних витрат, мінімізації ризиків виникнення помилкових рішень та забезпечення високих стандартів обслуговування клієнтів стимулює суб'єктів господарювання до впровадження сучасних інформаційних систем, зокрема ERP (Enterprise Resource Planning) та CRM (Customer Relationship Management) рішень. Однак, комплексні інформаційні системи подібного типу часто

характеризуються надмірним функціоналом або недостатнім рівнем адаптивності до специфічних особливостей бізнес-процесів конкретного підприємства. Зазначені обставини формують передумови для розробки адаптивних цифрових рішень, орієнтованих на конкретні потреби та галузеві особливості суб'єктів господарювання. Аналіз галузевих кейсів демонструє, що імплементація спеціалізованих програмних продуктів забезпечує суттєве підвищення ефективності управління виробничими замовленнями, скорочення тривалості виробничого циклу та оптимізацію ресурсних витрат [4].

1.2 Аналіз наявних інформаційних технологій предметної області

Дослідження сучасного інформаційно-технологічного інструментарію управління виробничими замовленнями передбачає комплексний аналіз функціональних можливостей, переваг та обмежень актуальних ERP- та CRM-систем, які використовуються для оптимізації процесів управління замовленнями, планування виробничої діяльності та автоматизації бізнес-процесів. Порівняльний аналіз існуючих технологічних рішень забезпечує можливість об'єктивної оцінки їх функціонального потенціалу, конкурентних переваг та обмежуючих факторів, що, в свою чергу, дозволяє обґрунтувати доцільність розробки власної інформаційної системи з високим рівнем адаптивності до специфіки конкретного підприємства [5].

ProShop ERP представляє собою веб-орієнтовану систему управління, призначену для малих та середніх машинобудівних підприємств, особливо тих, що функціонують у регульованих галузях, де вимагається відповідність стандартам ISO 9000, AS9100 тощо. Функціональна архітектура системи забезпечує реалізацію комплексного управління замовленнями та виробничими процесами, інтеграцію з CRM-компонентами, контроль якості та відповідності стандартам, а також управління інвентаризаційними процесами та ресурсним потенціалом підприємства. Серед ключових переваг ProShop

ERP доцільно виокремити спеціалізацію на процесах металообробки, високий рівень кастомізації системи відповідно до потреб конкретного підприємства, а також функціональні можливості забезпечення відповідності стандартам якості. Водночас, система характеризується певними обмеженнями, зокрема: надмірна складність для невеликих підприємств, необхідність значних часових витрат на впровадження та навчання персоналу. Аналіз функціональної відповідності ProShop ERP до потреб підприємств дозволяє стверджувати, що дана система найбільш релевантна для суб'єктів господарювання, які потребують детального контролю виробничих процесів та забезпечення відповідності високим стандартам якості [6].

MRPeasy позиціонується як хмарна ERP-система для малих та середніх виробничих підприємств, функціональний потенціал якої орієнтований на забезпечення ефективного управління виробництвом, інвентаризаційними процесами та фінансовими потоками. Система інтегрує функціональні модулі планування виробничої діяльності та управління замовленнями, забезпечення контролю інвентаризаційних процесів та постачань, фінансового обліку та формування звітності. Серед конкурентних переваг MRPeasy слід відзначити інтуїтивно зрозумілий інтерфейс, мінімізацію часових витрат на впровадження системи, а також доступну цінову політику для суб'єктів малого бізнесу. Обмеження системи включають недостатню функціональність для великих виробничих підприємств та необхідність придбання додаткових модулів для забезпечення специфічних потреб. Аналіз функціональної відповідності MRPeasy до завдань підприємств демонструє, що дана система є оптимальним вибором для суб'єктів господарювання, які орієнтовані на швидке та економічно доцільне впровадження базового функціоналу управління виробництвом та замовленнями [7].

Microsoft Dynamics 365 представляє собою комплексну ERP та CRM систему, яка характеризується високим рівнем інтегрованості з іншими продуктами Microsoft, зокрема Office 365 та Power BI. Функціональна архітектура системи включає модулі управління виробництвом та ланцюгами

постачань, фінансового обліку та аналітики, а також інтеграції з CRM-компонентами для забезпечення ефективного управління клієнтською базою. Конкурентні переваги Microsoft Dynamics 365 охоплюють широкий спектр можливостей налаштування системи, органічну інтеграцію з екосистемою Microsoft, а також підтримку як хмарних, так і локальних рішень. Обмежуючі фактори включають високу вартість впровадження та обслуговування системи, а також складність налаштування для забезпечення специфічних потреб підприємства. Аналіз функціональної відповідності Microsoft Dynamics 365 до завдань підприємств дозволяє констатувати, що дана система найбільш релевантна для середніх та великих суб'єктів господарювання, які потребують гнучкого та масштабованого рішення з широкими можливостями інтеграції [8].

Аналіз галузевої специфіки та функціональних особливостей наведених систем дозволяє констатувати наявність певних обмежень щодо їх використання на підприємствах з унікальними виробничими процесами. Зокрема, стандартизовані рішення часто не забезпечують повного охоплення специфічних бізнес-процесів, що генерує необхідність значної адаптації системи або модифікації існуючих процедур відповідно до функціональних можливостей програмного забезпечення. Крім того, вартісні параметри комплексних ERP-рішень можуть становити суттєвий бар'єр для малих та середніх підприємств, обмежуючи можливості цифрової трансформації виробничих процесів. Зазначені обставини актуалізують розробку спеціалізованих інформаційних систем з модульною архітектурою, що забезпечують можливість поетапного впровадження функціональних компонентів відповідно до потреб та фінансових можливостей підприємства [9].

Інтеграційний потенціал інформаційних систем управління виробничими замовленнями також заслуговує на особливу увагу в контексті забезпечення ефективної взаємодії з існуючими програмними рішеннями підприємства. Здатність системи до безшовної інтеграції з бухгалтерськими

програмами, системами управління складськими запасами, логістичними платформами та іншими компонентами інформаційної інфраструктури суттєво впливає на ефективність цифрової трансформації бізнес-процесів. Відповідно, при розробці власної інформаційної системи управління виробничими замовленнями доцільно враховувати необхідність забезпечення високого рівня інтеграційної взаємодії з існуючими програмними рішеннями, що дозволить мінімізувати дублювання даних та оптимізувати інформаційні потоки в межах підприємства [10].

Безпекові аспекти функціонування інформаційних систем управління виробничими замовленнями набувають особливої актуальності в контексті підвищення рівня кіберзагроз та посилення нормативних вимог щодо захисту даних. Сучасні системи повинні забезпечувати належний рівень захисту конфіденційної інформації клієнтів, технологічних даних та фінансових показників підприємства. Аналіз функціональних характеристик наведених ERP-систем демонструє різний рівень імплементації безпекових механізмів, що обумовлює необхідність ретельної оцінки відповідних параметрів при виборі програмного рішення або розробці власної інформаційної системи [11].

Еволюційні тенденції розвитку інформаційних систем управління виробничими замовленнями свідчать про поступову інтеграцію елементів штучного інтелекту та машинного навчання, що забезпечує підвищення точності прогнозування, оптимізацію планування виробництва та автоматизацію рутинних процесів. Зокрема, алгоритми машинного навчання використовуються для прогнозування термінів виконання замовлень на основі історичних даних, оптимізації розподілу ресурсів та ідентифікації потенційних ризиків у виробничому процесі. Впровадження подібних технологій дозволяє суттєво підвищити ефективність управління виробничими замовленнями та забезпечити конкурентні переваги підприємства на ринку [12].

На основі проведеного аналізу можна констатувати, що сучасний ринок інформаційних систем управління виробничими замовленнями пропонує

широкий спектр рішень різного рівня складності та функціональності. Однак, вибір оптимального програмного забезпечення або рішення щодо розробки власної інформаційної системи повинен базуватися на комплексній оцінці специфіки бізнес-процесів підприємства, масштабів його діяльності, фінансових можливостей та стратегічних цілей розвитку. Критерії вибору інформаційної системи повинні включати: функціональну відповідність специфічним потребам підприємства, масштабованість рішення, можливості інтеграції з існуючими програмними продуктами, вартісні параметри впровадження та обслуговування, рівень технічної підтримки та навчання персоналу [13].

Таблиця 1.1 – Представлення порівняльної таблиці існуючих рішень

Параметр	**ProShop ERP**	**MRPeasy**	**Microsoft Dynamics 365**
1	**2**	**3**	**4**
Тип	ERP + MES для виробництва	Хмарна ERP-система	Комплексна ERP + CRM
Основний функціонал	Управління замовленнями, виробництвом, якістю	Управління виробництвом, інвентарем, фінансами	Виробництво, облік, CRM, аналітика
Інтерфейс	Веб-інтерфейс, технічний	Простий, інтуїтивно зрозумілий	Багатофункціональний, складний для новачків
Переваги	Глибока спеціалізація, контроль якості	Доступність, простота, швидке впровадження	Широкі можливості інтеграції, аналітика
Недоліки	Складність впровадження, потребує навчання	Обмежена гнучкість для складних задач	Висока вартість, складність адаптації
Доступність	Хмарна / локальна	Хмарна	Хмарна / локальна

Продовження табл. 1.1

1	2	3	4
Масштабованість	Середній та великий бізнес	Малий та середній бізнес	Підприємства будь-якого масштабу
Підтримка ШІ	Частково (аналітика, автоматизація)	Обмежена	Так (Power BI, Copilot AI)
Відповідність задачам проєкту	Частково	Частково	Частково (залежить від кастомізації)
Цінова політика	Висока	Середня / доступна	Висока

Сучасний ландшафт інформаційно-технологічних рішень у сфері управління виробничими замовленнями характеризується значною диверсифікацією функціональних можливостей, архітектурних особливостей та імплементаційних підходів. Компаративний аналіз репрезентативних програмних рішень – ProShop ERP, MRPeasy та Microsoft Dynamics 365 – дозволяє виявити їх структурно-функціональні особливості, диференціювати сфери оптимального застосування та ідентифікувати потенційні обмеження у контексті специфічних виробничих процесів [5].

ProShop ERP представляє собою інтегрований програмний комплекс, архітектурно позиціонований на перетині традиційних систем планування ресурсів підприємства (ERP) та систем виконання виробничих процесів (MES), що забезпечує підвищену релевантність рішення для промислових підприємств. Функціональна архітектура системи забезпечує інтеграцію процесів управління виробництвом, закупівельною діяльністю та фінансовими потоками у єдиному інформаційному просторі. Технологічна платформа ProShop ERP, з високою вірогідністю, базується на серверних технологіях високої надійності, що забезпечують стабільність функціонування та безперебійність доступу до даних через веб-інтерфейс. Програмна архітектура системи, імовірно, включає серверний компонент на основі Java або C#, що дозволяє розробити потужні алгоритми обробки складноструктурованих

виробничих даних. Репозиторій даних, з високою долею вірогідності, базується на реляційній системі управління базами даних, такій як PostgreSQL або Microsoft SQL Server, що забезпечує структурну цілісність та транзакційну безпеку інформаційних масивів. Інтерфейсна частина системи, орієнтована на технологічну спеціалізацію користувачів, реалізована на основі стандартизованих веб-технологій (HTML/CSS/JavaScript), потенційно з використанням React або інших фреймворків для забезпечення динамічності та інтерактивності користувацького досвід.

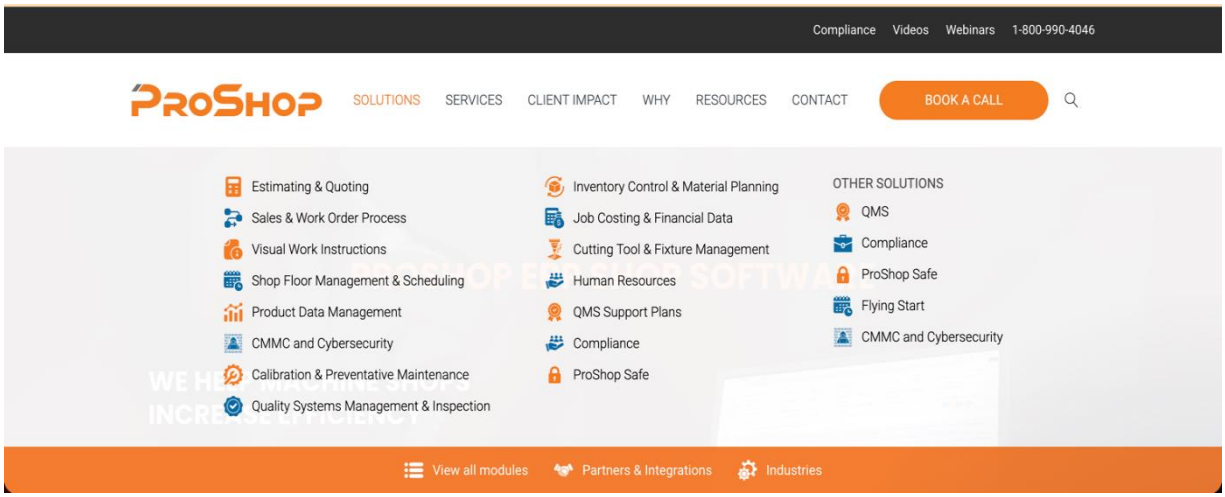


Рисунок 1.1. Загальна архітектура та ключові компоненти ProShop ERP

Інформаційний контекст замовлення в системі ProShop ERP характеризується комплексністю та детальною структурованістю, що відображено на рисунку 1.2. Система забезпечує акумулювання та систематизацію усіх релевантних параметрів замовлення, включаючи технічні специфікації, часові обмеження, фінансові умови та додаткові вимоги клієнта. Інтеграція даних у єдиному інформаційному просторі дозволяє мінімізувати ризики інформаційних втрат та підвищити точність техніко-економічних розрахунків [7].

Дане фото описує наявні можливості та функції системи

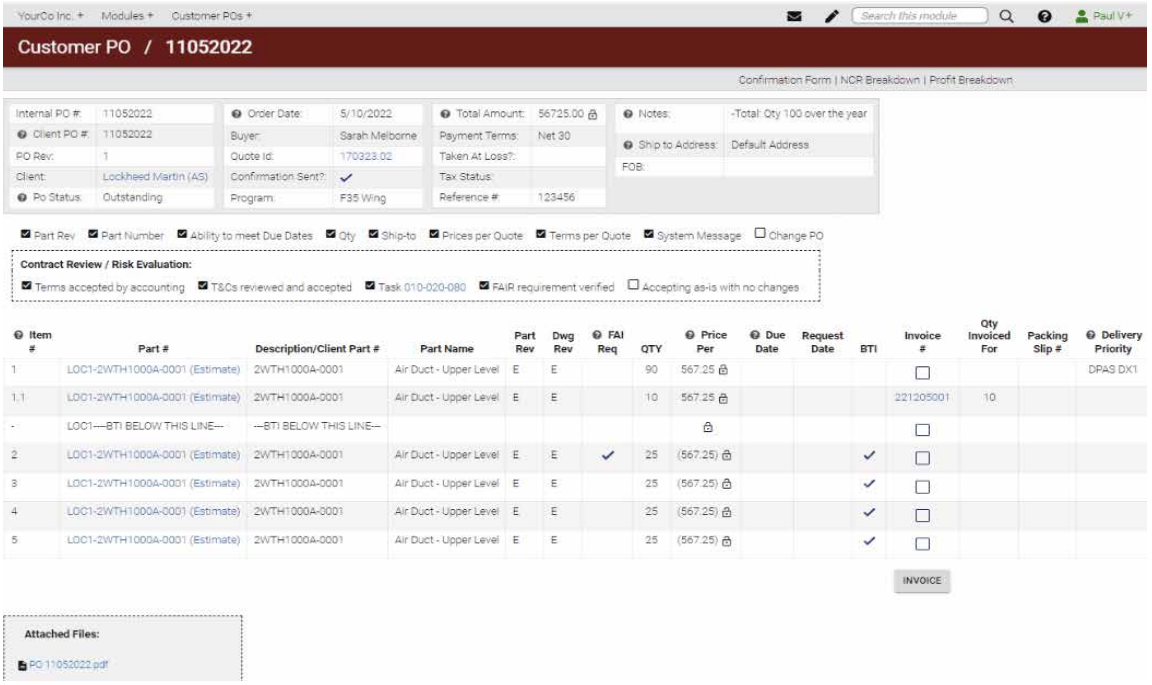


Рисунок 1.2 Інформація щодо замовлення користувача

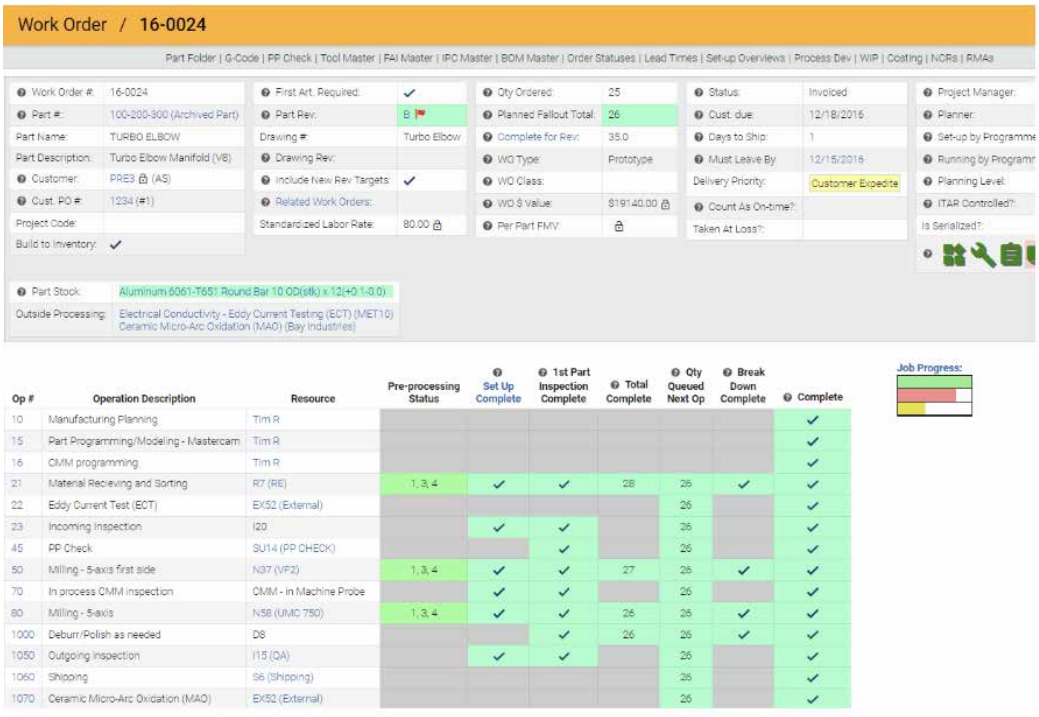


Рисунок 1.3 Робочі процеси системи

MRPeasy: Як хмарна ERP-система для малого та середнього бізнесу, MRPeasy, ймовірно, створена з акцентом на легкість і доступність. Для бекенду могли обрати гнучкі мови, такі як Python або PHP, які ідеально

підходять для хмарних рішень, а для фронтенду — JavaScript із сучасними бібліотеками, наприклад Vue.js або React, щоб забезпечити інтуїтивний інтерфейс. База даних, швидше за все, побудована на основі PostgreSQL або MySQL, із розгортанням у хмарі, можливо, на платформах типу AWS чи Google Cloud, що дозволяє швидко масштабувати систему. Автоматизація прорахунків матеріалів, ймовірно, реалізована через REST API, що забезпечує гнучку взаємодію між компонентами.

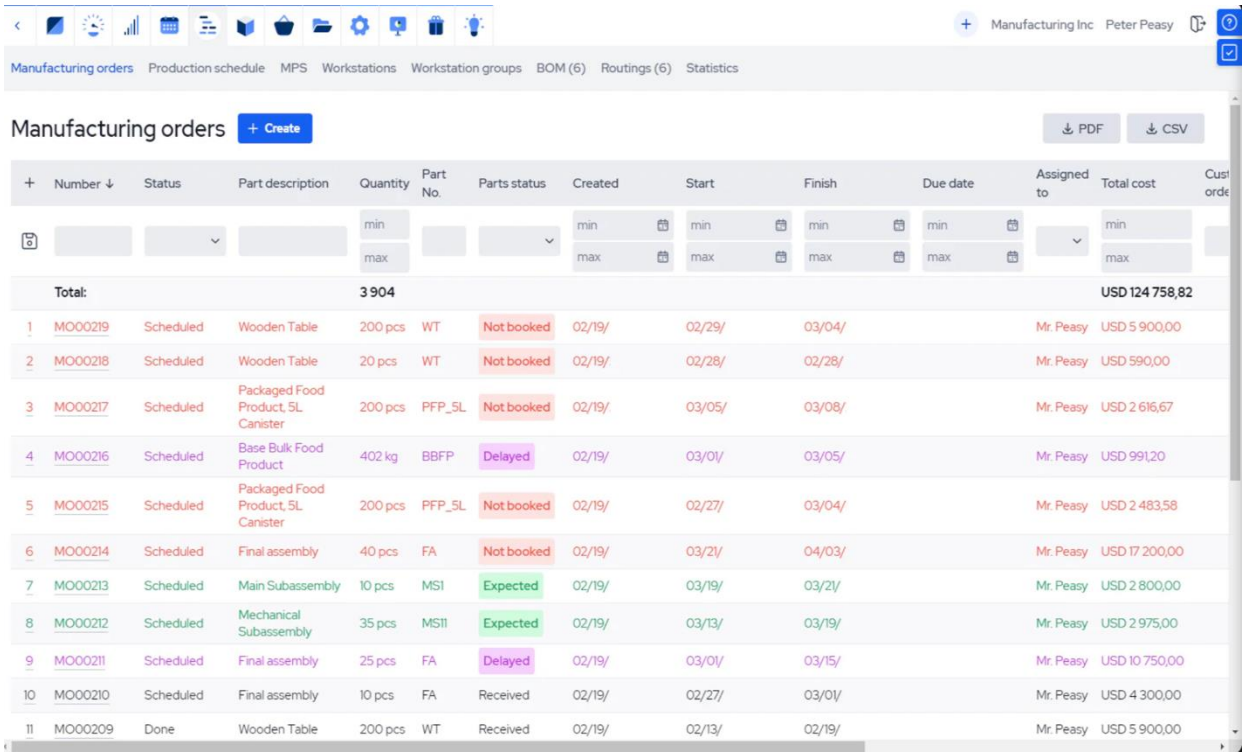


Рисунок 1.4 Опис мануфактури

Microsoft Dynamics 365: Ця система, створена технологічним гігантем Microsoft, безсумнівно, базується на їхньому власному екосистемному підході. Бекенд, швидше за все, побудований на платформі .NET із використанням C#, із розгортанням у хмарі Azure для максимальної масштабності. Для зберігання даних могли бути використані Microsoft SQL Server для структурованих даних та Cosmos DB для гнучких NoSQL-структур. Фронтенд, ймовірно, створений із застосуванням TypeScript та фреймворків типу Angular або React, що забезпечує сучасний і динамічний інтерфейс. Аналітичні можливості, такі як

Power BI чи Copilot AI, інтегровані через REST API та Graph API, додаючи інноваційні інструменти для аналізу та автоматизації.

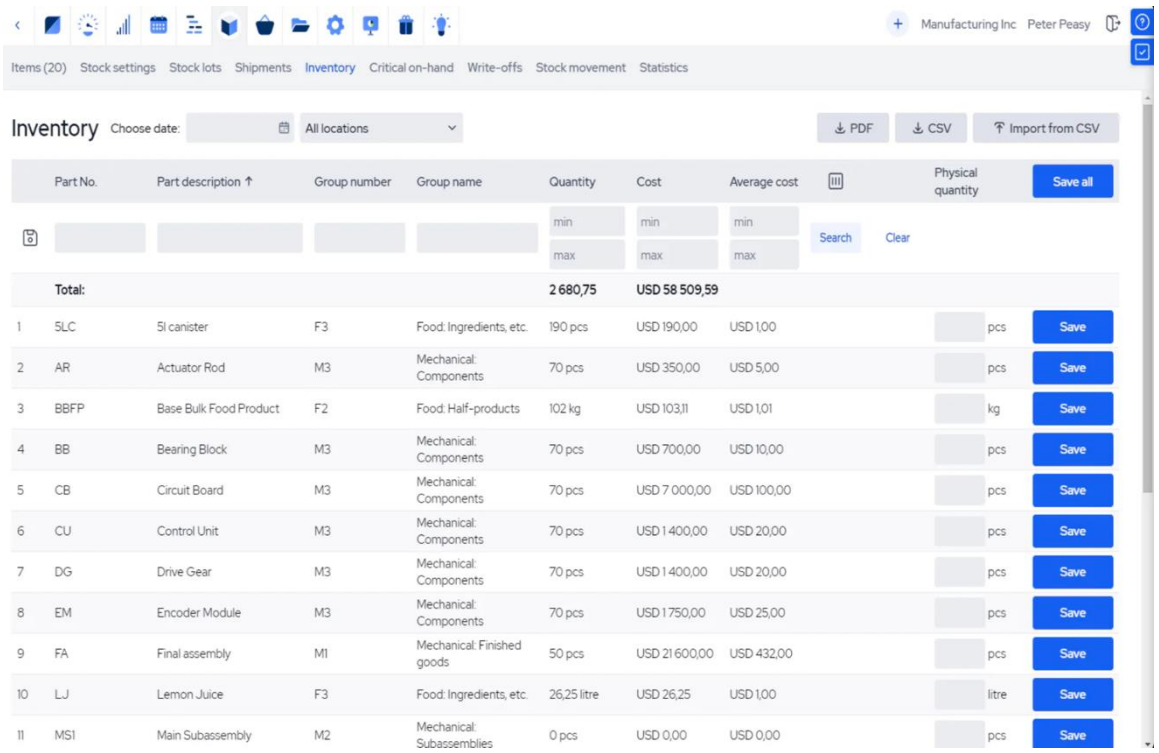


Рисунок 1.5 Інвентаризція в системі

Системи демонструють різні підходи до архітектури: ProShop використовує локальну інтеграцію з виробництвом, MRPeasy — хмарну легкість, а Dynamics 365 — масштабність із хмарними сервісами. Моя система комбінує ці принципи, використовуючи Angular для фронтенду, Spring Boot для бекенду та MongoDB для даних, що забезпечує гнучкість і масштабованість.

1.3 Визначення вимог до інструментальних засобів

Формування вимог до інструментальних засобів розробки веб-системи управління замовленнями для підприємств металообробної галузі базується на комплексному аналізі функціональних потреб, технологічних обмежень та стратегічних цілей бізнес-структур даного сектору. Концептуалізація веб-

системи управління замовленнями детермінується необхідністю комплексної оптимізації внутрішніх бізнес-процесів, мінімізації часових витрат на виконання рутинних операцій, підвищення продуктивності праці персоналу та забезпечення транспарентності й контрольованості усіх етапів життєвого циклу замовлення [16].

Фундаментальною вимогою до розроблюваної системи є забезпечення автоматизації ключових бізнес-процесів, пов'язаних з управлінням замовленнями – від етапу ініціації до фінального виконання, що дозволяє мінімізувати вплив суб'єктивних факторів, знизити ризик виникнення помилок, оптимізувати інформаційні потоки між структурними підрозділами та скоротити часові витрати на обробку технічної документації. Архітектурна концепція системи повинна забезпечувати диференційований доступ користувачів з різними функціональними ролями, що підвищує ефективність розподілу завдань між працівниками та оптимізує загальну операційну ефективність підприємства [17].

Технологічний фундамент розроблюваної системи повинен базуватися на сучасних веб-технологіях, що забезпечують кросплатформену доступність, інтерактивність користувацького досвіду та гнучкість масштабування. Аналіз технологічних тенденцій та практичного досвіду імплементації аналогічних систем свідчить про доцільність використання архітектурної парадигми Single-Page Application (SPA) з використанням фреймворку Angular для реалізації клієнтської частини системи. Angular забезпечує необхідний рівень модульності, компонентну структуру та двосторонню прив'язку даних, що оптимізує процес розробки складних інтерфейсів та забезпечує позитивний користувацький досвід [18].

Серверна частина системи потребує імплементації на основі надійних та масштабованих технологій, серед яких оптимальним вибором видається фреймворк Spring Boot, що забезпечує високий рівень абстракції, інтеграційну гнучкість та широкі можливості для реалізації бізнес-логіки. Архітектурний підхід на основі мікросервісів дозволяє забезпечити модульність та

масштабованість системи, що особливо важливо в контексті потенційного розширення функціональності або адаптації до змінних бізнес-потреб підприємства [19].

Репозиторій даних розроблюваної системи доцільно реалізувати на основі нереляційної бази даних MongoDB, що забезпечує гнучкість схеми даних, високу продуктивність при операціях читання/запису та горизонтальну масштабованість. Документо-орієнтована парадигма MongoDB дозволяє ефективно моделювати складноструктуровані дані, характерні для виробничих замовлень, з можливістю динамічної модифікації структури без необхідності перебудови схеми бази даних [20].

Функціональні вимоги до системи управління замовленнями

Функціональні вимоги до розроблюваної системи охоплюють комплекс модулів, спрямованих на всебічну автоматизацію процесів обробки виробничих замовлень. Першочерговий аспект функціональності стосується прийому замовлень, що передбачає імплементацію механізмів ручного створення карток замовлень через інтерфейс користувача, а також автоматизований імпорт даних із зовнішніх джерел, зокрема електронних таблиць Excel. Функціональний модуль повинен забезпечувати багатостадійну обробку замовлень з можливістю відстеження поточного статусу та акумулювання детальної інформації про клієнта, включаючи контактні дані, історію взаємодії та преференції щодо умов співпраці [21].

Технічний прорахунок замовлення становить наступний функціональний блок, який потребує реалізації алгоритмів напівавтоматичної калькуляції матеріальних та ресурсних потреб для виконання конкретного замовлення. Даний модуль повинен забезпечувати прогностичний аналіз вартісних параметрів виробництва з урахуванням матеріалоємності, трудомісткості та енергоємності виробничих процесів. Інтелектуальна складова модуля має формувати рекомендації щодо оптимального вибору технологічного оснащення та виробничого обладнання, базуючись на

параметрах замовлення та наявній технічній інфраструктурі підприємства [22].

Функціональний модуль прорахунку ресурсів та перевірки наявності повинен забезпечувати інтеграцію з системами управління складськими запасами для оперативної верифікації доступності необхідних матеріалів. У випадку дефіциту ресурсів система має генерувати структурований перелік відсутніх матеріалів з можливістю автоматичного формування закупівельних заявок. Даний функціональний блок є критичним для забезпечення безперервності виробничого процесу та оптимізації логістичних ланцюгів підприємства [23].

Рекомендаційний модуль по виробництву повинен реалізовувати алгоритми автоматичного визначення оптимальних виробничих технологій на основі параметрів замовлення, доступних ресурсів та технологічних можливостей підприємства. Функціональність даного блоку включає планування послідовності виробничих етапів з визначенням часових рамок, ресурсного забезпечення та контрольних точок для моніторингу якості. Інтелектуальна оптимізація виробничих процесів дозволяє мінімізувати часові та матеріальні витрати при забезпеченні високих стандартів якості продукції [24].

Бухгалтерський прорахунок, як функціональний модуль системи, повинен забезпечувати комплексне обчислення вартісних параметрів замовлення з урахуванням прямих та непрямих витрат, податкових зобов'язань та очікуваної маржинальності. Даний блок має реалізовувати функціональність автоматичного генерування рахунків та інтеграції з платіжними системами для забезпечення оперативного фінансового обслуговування клієнтів. Автоматизація фінансових розрахунків мінімізує ризики помилок та оптимізує процеси фінансового обліку підприємства [25].

Мультирольовий інтерфейс користувача повинен забезпечувати диференційований доступ до функціональності системи відповідно до посадових обов'язків та рівня відповідальності користувачів.

Адміністративний рівень доступу передбачає повний контроль над конфігурацією системи та управління користувачами; менеджерський рівень фокусується на управлінні замовленнями та взаємодії з клієнтами; технічний рівень забезпечує доступ до функціональності прорахунку та планування виробництва; бухгалтерський рівень орієнтований на фінансові аспекти обробки замовлень. Диференціація доступу підвищує безпеку системи та оптимізує користувацький досвід відповідно до професійних потреб персоналу [26].

Функціональний модуль звітності повинен забезпечувати генерування аналітичних звітів різного рівня деталізації з можливістю експорту у популярні формати даних, зокрема Excel та PDF. Система має підтримувати як стандартизовані шаблони звітів, так і можливість гнучкого налаштування параметрів аналітики відповідно до потреб користувачів. Аналітичний потенціал даного модуля є критичним для забезпечення інформаційної підтримки управлінських рішень та оптимізації бізнес-процесів підприємства [27].

Нефункціональні вимоги до системи управління замовленнями

Нефункціональні вимоги до розроблюваної системи детермінують якісні характеристики програмного продукту, які суттєво впливають на ефективність його експлуатації та задоволеність користувачів. Якісні параметри системи включають вимоги щодо надійності функціонування з показником доступності не менше 99,5% часу, що є критичним для забезпечення безперервності бізнес-процесів підприємства. Архітектурні рішення повинні забезпечувати високу доступність системи через механізми резервування, балансування навантаження та автоматичного відновлення після збоїв [28].

Ергономічні характеристики інтерфейсу користувача є фундаментальною нефункціональною вимогою, що визначає практичну цінність системи для кінцевих користувачів. Інтерфейс повинен характеризуватися інтуїтивною зрозумілістю, логічною структурованістю та

мінімізацією когнітивного навантаження на користувача. Імплементація мультирольового доступу має забезпечувати адаптацію інтерфейсу відповідно до функціональних потреб та рівня технічної компетентності різних категорій користувачів [29].

Безпекові аспекти функціонування системи охоплюють комплекс механізмів захисту інформації, включаючи багаторівневу автентифікацію користувачів, авторизацію доступу до функціональних модулів, шифрування даних при передачі та зберіганні, а також систему контролю та аудиту дій користувачів. Реалізація безпекових механізмів повинна забезпечувати баланс між захищеністю даних та зручністю користувацького досвіду, що є критичним для практичної імплементації системи [30].Перформативні характеристики системи детермінуються вимогами щодо швидкодії обробки запитів з максимальним часом відгуку не більше 3-5 секунд для типових операцій. Архітектурні рішення повинні забезпечувати лінійну масштабованість системи при збільшенні кількості користувачів та обсягу даних, що особливо актуально для підприємств з динамічним розвитком бізнесу. Оптимізація алгоритмів обробки даних та кешування часто використовуваної інформації дозволяє забезпечити високу швидкодію системи навіть при значному навантаженні [31].

Інтеграційні можливості та підтримка є критичними нефункціональними вимогами, що визначають довгострокову цінність системи для підприємства. Архітектура повинна забезпечувати гнучкі механізми інтеграції з існуючими інформаційними системами, включаючи бухгалтерське програмне забезпечення, системи управління складськими запасами та логістичні платформи. Документаційний супровід системи має включати детальні інструкції для користувачів різних рівнів, технічну документацію для адміністраторів та розробників, а також методичні рекомендації щодо інтеграції та масштабування [32].

Мультимовність інтерфейсу користувача є важливою вимогою для підприємств, що працюють на міжнародних ринках або мають

мультинаціональний персонал. Система повинна забезпечувати можливість легкого перемикання між мовами інтерфейсу без необхідності перезапуску або перелогінювання. Архітектурні рішення мають передбачати екстерналізацію текстових ресурсів для спрощення процесу локалізації та розширення мовної підтримки [33].

Кросплатформенна доступність системи визначається вимогами щодо підтримки основних веб-браузерів, включаючи Google Chrome, Mozilla Firefox, Apple Safari та Microsoft Edge. Адаптивний дизайн інтерфейсу повинен забезпечувати комфортне використання системи на різних типах пристроїв, від десктопних комп'ютерів до мобільних телефонів, що розширює можливості доступу до системи для працівників, які перебувають поза офісом або виробничим приміщенням [34].

Комплексна інтеграція функціональних та нефункціональних вимог у рамках єдиної архітектурної концепції дозволяє сформувати технічне завдання на розробку веб-системи управління замовленнями для підприємств металообробної галузі, що забезпечить оптимальний баланс між технологічною складністю, функціональною повнотою та економічною ефективністю імплементації [35].

Висновки до розділу 1

Проведений теоретико-методологічний аналіз систем управління виробничими замовленнями дозволяє сформулювати ряд важливих висновків, що закладають фундамент для подальшого дослідження та практичної розробки веб-системи управління замовленнями для підприємств металообробної галузі.

Комплексне дослідження предметної області виявило, що управління виробничими замовленнями є критичним компонентом операційної діяльності виробничих підприємств, який суттєво впливає на їх конкурентоспроможність, прибутковість та ринкові позиції. Аналіз сучасного

стану галузі демонструє, що значна кількість підприємств продовжує використовувати застарілі або фрагментовані інструменти управління замовленнями, що призводить до інформаційної розрізненості, дублювання даних, підвищеної залежності від людського фактору та зниження ефективності бізнес-процесів.

Структурно-функціональна декомпозиція процесу управління замовленнями виявила п'ять ключових компонентів: система прийому замовлень, техніко-технологічний аналіз, фінансово-економічний аналіз, планування виробництва та система звітності. Кожен компонент потребує відповідного інформаційно-технологічного забезпечення для оптимізації процесів та мінімізації ризиків помилок. Інтеграція цих компонентів у єдиному інформаційному просторі є критичною для забезпечення узгодженості та ефективності виробничого циклу.

Аналіз існуючих інформаційних технологій у сфері управління виробничими замовленнями дозволив ідентифікувати три репрезентативні системи – ProShop ERP, MRPeasy та Microsoft Dynamics 365, які демонструють різні підходи до автоматизації відповідних бізнес-процесів. Компаративний аналіз функціональності, архітектури та цінової політики цих систем виявив, що кожна з них має свої переваги та обмеження, які детермінують їх оптимальне застосування для підприємств різного масштабу та галузевої специфіки.

ProShop ERP демонструє глибоку спеціалізацію у сфері металообробки та машинобудування з акцентом на контроль якості та відповідність галузевим стандартам, проте характеризується підвищеною складністю та значними витратами на впровадження. MRPeasy пропонує доступне хмарне рішення з інтуїтивно зрозумілим інтерфейсом, проте має обмежену функціональність для великих виробництв. Microsoft Dynamics 365 забезпечує широкі можливості інтеграції та аналітики, проте вимагає значних інвестицій та має складний процес адаптації до специфічних потреб підприємства.

Визначення вимог до інструментальних засобів розробки веб-системи управління замовленнями для підприємств металообробної галузі дозволило сформувати комплексне бачення функціональних та нефункціональних характеристик майбутньої системи. Функціональні вимоги охоплюють сім ключових модулів: прийом замовлень, технічний прорахунок, прорахунок ресурсів, рекомендації по виробництву, бухгалтерський прорахунок, мультирольовий інтерфейс та систему звітності. Нефункціональні вимоги включають якісні параметри надійності, ергономічності, безпеки, швидкодії, інтеграційних можливостей, мультимовності та кросплатформенної доступності.

Технологічний фундамент розроблюваної системи доцільно базувати на сучасних веб-технологіях з використанням архітектурної парадигми Single-Page Application на основі фреймворку Angular для клієнтської частини, фреймворку Spring Boot для серверної частини та нереляційної бази даних MongoDB для зберігання даних. Даний технологічний стек забезпечує оптимальний баланс між продуктивністю, масштабованістю та економічною ефективністю розробки.

Комплексна інтеграція виявлених функціональних та нефункціональних вимог у рамках єдиної архітектурної концепції дозволяє сформулювати технічне завдання на розробку веб-системи управління замовленнями, що забезпечить оптимізацію бізнес-процесів металообробних підприємств, підвищення продуктивності праці персоналу, зниження операційних витрат та підвищення рівня обслуговування клієнтів. Практична імплементація такої системи становить перспективний напрямок для подальшого дослідження та розробки.

Проведений аналіз створює методологічне підґрунтя для переходу до наступного етапу дослідження – проектування архітектури та розробки програмного забезпечення веб-системи управління замовленнями для підприємств металообробної галузі, що становитиме предмет розгляду в наступних розділах роботи. Концептуальне бачення системи, сформоване на

основі теоретико-методологічного аналізу, забезпечує цілісність та логічну послідовність подальшого дослідження.

РОЗДІЛ 2 АРХІТЕКТУРНО-ТЕХНОЛОГІЧНІ АСПЕКТИ ПРОЄКТУВАННЯ ВЕБ-СИСТЕМИ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ

2.1 Опис загальної концепції системи

2.1.1 Аналіз предметної області та постановка задачі

Виробничі підприємства металообробної галузі функціонують в умовах високої конкуренції та динамічних змін ринкового середовища, що вимагає оптимізації внутрішніх бізнес-процесів для забезпечення конкурентоспроможності та підвищення економічної ефективності. Ключовою функціональною складовою діяльності металообробних підприємств виступає процес управління замовленнями, що характеризується багатоетапністю та мультисуб'єктністю. Аналіз операційної діяльності суб'єктів господарювання даного сегменту демонструє, що стандартизований цикл обробки виробничого замовлення охоплює низку послідовних етапів, включаючи акумулювання первинної інформації про клієнтські потреби, проведення техніко-технологічної оцінки виробничих можливостей, фінансово-економічний прорахунок, безпосередньо виробничий процес та логістично-сервісний супровід.

Практичне впровадження зазначеного циклу супроводжується низкою системних проблем, серед яких доцільно виокремити: недостатню автоматизацію процесів ручного введення даних, що генерує високий ризик виникнення помилок та неточностей; фрагментарність інформаційних потоків, що призводить до втрати даних або їх дублювання; недостатній рівень координації між функціональними підрозділами підприємства, зокрема між відділами продажу, технологічним та бухгалтерським блоками; відсутність централізованої системи моніторингу статусу виконання замовлень, що ускладнює контроль та управління виробничими процесами. Зазначені проблеми актуалізують необхідність розробки та впровадження

спеціалізованої веб-системи управління замовленнями, яка забезпечить комплексну автоматизацію та оптимізацію відповідних бізнес-процесів.

Архітектурно-функціональна концепція веб-системи базується на мультирольовому підході, що передбачає диференційований доступ користувачів до функціональності відповідно до їх організаційно-посадових обов'язків. Структура користувачів системи відображена у таблиці 2.1, де ідентифіковано чотири ключові ролі з відповідними функціональними повноваженнями.

Таблиця 2.1 – Структура користувачів системи

Роль	Опис	Повноваження
1	**2**	**3**
Адміністратор	Користувач із повним контролем над системою, який виконує функції системного адміністрування та загального управління інформаційними потоками	- Управління обліковими записами користувачів Редагування будь-яких даних Перегляд усіх замовлень Повний доступ до всіх модулів системи
Менеджер	Основна контактна особа з клієнтами, відповідальна за прийом, створення і контроль замовлень на всіх етапах життєвого циклу	- Створення замовлень Контроль виконання Перегляд інформації по замовленнях Майже ідентичний доступ до функцій, як у адміністратора
Технолог	Спеціаліст, відповідальний за техніко-технологічну оцінку виробничих можливостей та ресурсних потреб для реалізації замовлення	- Доступ до технологічного блоку замовлень Проведення технічного прорахунку Обмежений доступ до інших частин системи

Продовження таблиці 2.1

1	2	3
Бухгалтер	Фахівець, відповідальний за фінансово-економічні аспекти обробки замовлень, включаючи калькуляцію собівартості та визначення кінцевої вартості	- Проведення фінансового прорахунку Доступ до фінансових даних Відсутність доступу до технічної або загальної інформації замовлення

Для забезпечення візуалізації функціональних взаємозв'язків між користувачами системи та їх взаємодії з програмним комплексом розроблено діаграму варіантів використання (Use Case Diagram), представлену на рисунку 2.1. Діаграма демонструє чітке розмежування функціональних ролей та відповідних прав доступу в рамках системи управління замовленнями.

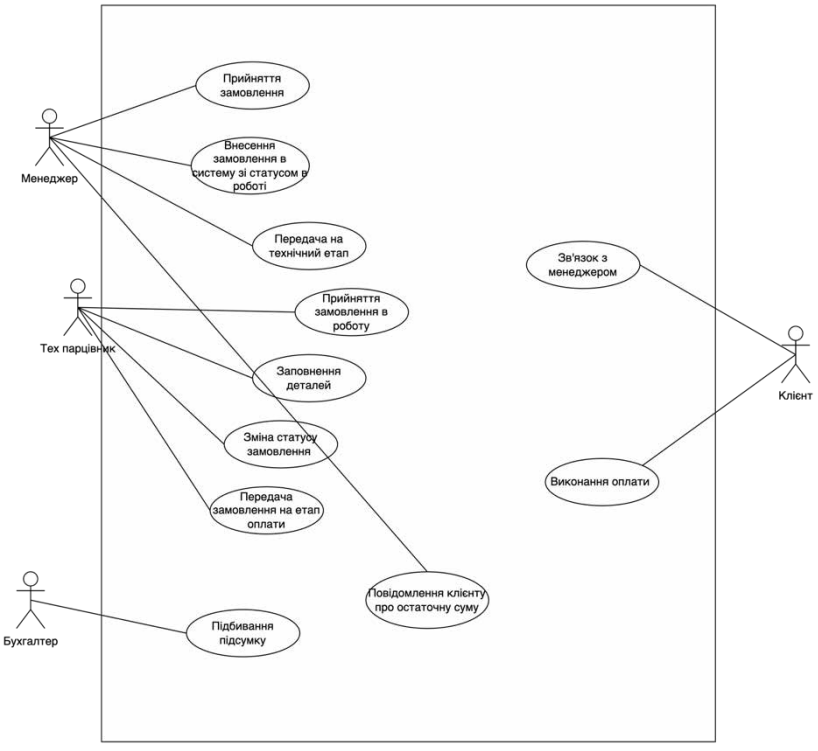


Рисунок 2.1 – Діаграма варіантів використання веб-системи управління замовленнями

На діаграмі візуалізовано ієрархію доступу до функціональності системи, де адміністратор має повний доступ до всіх модулів, менеджер

відповідає за створення та супровід замовлень, технолог фокусується на техніко-технологічних аспектах замовлення, а бухгалтер відповідає за фінансово-економічний блок. Кожна роль взаємодіє з системою через визначені випадки використання, що забезпечує логічну структуризацію функціональних обов'язків та мінімізує ризики несанкціонованого доступу до даних.

2.1.2 Бізнес-логіка обробки замовлення

Процесуальний алгоритм обробки замовлення в рамках веб-системи структурований за принципом послідовної передачі відповідальності між функціональними суб'єктами та реалізується у чотирьох ключових етапах.

Етап 1: Створення замовлення (відповідальний – Менеджер). Ініціація процесу відбувається при зверненні клієнта через доступні комунікаційні канали (телефонний зв'язок, електронна пошта, особисте звернення), на основі чого менеджер формує нове замовлення в системі. На даному етапі заповнюється базовий інформаційний комплекс, включаючи контактні дані клієнта, специфікацію продукції, часові обмеження та додаткові вимоги. Після збереження інформації замовлення автоматично передається на наступний етап – технічний прорахунок.

Етап 2: Технічний прорахунок (відповідальний – Технолог). На основі первинних даних, введених менеджером, технолог здійснює комплексний розрахунок матеріальних потреб, підбір оптимального виробничого обладнання та визначення технологічної схеми виготовлення продукції. Завершення введення всіх технічних параметрів ініціює автоматичну передачу замовлення на етап бухгалтерського прорахунку.

Етап 3: Фінансовий прорахунок (відповідальний – Бухгалтер). Бухгалтер формує кінцеву вартість замовлення на основі комплексного аналізу даних, введених менеджером і технологом. Фінансовий прорахунок включає калькуляцію вартості матеріалів, трудових ресурсів, амортизаційних

відрахувань, логістичних витрат та комерційної маржі. Завершення даного етапу переводить замовлення у статус сформованого та інтегрує його до загального переліку активних замовлень.

Етап 4: Управління замовленнями. Всі активні замовлення, що пройшли повний цикл прорахунків, відображаються у загальній таблиці замовлень. Паралельно, новостворені замовлення, що знаходяться на проміжних етапах обробки, доступні у спеціалізованій вкладці «Нові замовлення», що забезпечує ефективний моніторинг статусів та черговості обробки.

Багатостадійність процесу обробки замовлень вимагає чіткої системи статусного позиціонування, що реалізовано через таблицю статусів, представлену у таблиці 2.2. Кожен статус має унікальний код та відповідає певному етапу обробки замовлення, що дозволяє забезпечити транспарентність процесу та ефективний контроль за рухом замовлень у системі.

Таблиця 2.2 – Таблиця статусів замовлень

Код статусу	Опис	Відповідальна роль
WAITING	Замовлення щойно створено менеджером і ще не пройшло технічну оцінку	Менеджер
COUNTING	Технолог виконує оцінку необхідних ресурсів, матеріалів, обладнання	Технолог
SENDING	Завершено технічний етап, дані передано бухгалтеру	Бухгалтер
NON_PAID	Фінансовий прорахунок завершено, система очікує оплату замовлення від клієнта	Клієнт / Бухгалтер
PAID	Оплату підтверджено, замовлення передано у виробництво або вважається активним	Система / Менеджер
SET_OFF	Оплату внесено вручну (наприклад, при банківському переказі або ручній перевірці платежу)	Бухгалтер / Адміністратор

Функціональна ефективність системи управління замовленнями значною мірою залежить від адекватної категоризації замовлень, що дозволяє

оптимізувати процеси обробки та пріоритезації. Розроблена класифікаційна структура замовлень, представлена у таблиці 2.3, забезпечує багатовимірну категоризацію за типами виконання, клієнтської приналежності, пріоритетності, стану готовності даних та типу продукції.

Таблиця 2.3 – Класифікація замовлень у системі

Категорія класифікації	Тип / Підкатегорія	Опис
Тип виконання	Індивідуальне	Замовлення створене під конкретного клієнта, враховує нестандартні вимоги
	Серійне	Замовлення стосується партії однотипної продукції, що виготовляється за стандартом
Тип клієнта	Фізична особа	Замовлення надане приватною особою, зазвичай у малих обсягах
	Юридична особа	Замовлення від компанії чи організації, часто з великими обсягами чи повторюваними партіями
Пріоритетність	Звичайне	Виконується у стандартному режимі, згідно з чергою
	Термінове	Має високий пріоритет і виконується швидше за рахунок внутрішньої оптимізації
Стан готовності даних	Повне	Усі необхідні технічні й фінансові дані вже наявні
	Неповне	Потрібно уточнення або додаткова інформація для прорахунку чи виготовлення
Тип продукції	Стандартна	Продукт відповідає типовому каталогу виробів підприємства
	Спеціалізована	Виріб виготовляється за унікальним кресленням або специфікацією клієнта

2.1.3 Проблемні місця, які вирішує система

Впровадження веб-системи управління замовленнями спрямоване на вирішення низки системних проблем, характерних для традиційних підходів

до управління виробничими процесами на металообробних підприємствах. Ключовими «проблемними місцями», які усуваються завдяки розробленій системі, є зменшення ризику втрати або дублювання інформації через централізацію даних у єдиній системі, що забезпечує надійне зберігання, версійність та контроль доступу до інформації про замовлення.

Автоматизація прорахунків та зниження впливу людського фактору реалізується шляхом впровадження алгоритмів автоматичного розрахунку ресурсних потреб, технологічних параметрів та фінансових показників, що мінімізує ризик виникнення помилок та підвищує точність прогнозування.

Покращення комунікації між функціональними підрозділами (менеджмент, технологічний блок, фінансово-економічна служба) досягається через інтеграцію всіх етапів обробки замовлення у єдиному інформаційному просторі з чітким розмежуванням відповідальності та автоматизованою передачею даних між суб'єктами процесу.

Централізований доступ до історії замовлень та аналітичної інформації забезпечує можливість ретроспективного аналізу, виявлення паттернів, прогнозування тенденцій та оптимізації управлінських рішень на основі накопичених даних.

Чітке розмежування прав доступу між функціональними ролями підвищує рівень інформаційної безпеки, запобігає несанкціонованому доступу до даних та забезпечує персоналізацію відповідальності за конкретні етапи обробки замовлення.

2.1.4 Потенційні проблемні місця при використанні веб-системи управління замовленнями

Незважаючи на значні переваги, впровадження та експлуатація веб-системи управління замовленнями може супроводжуватися певними викликами та «проблемними місцями», які потребують превентивного аналізу та розробки стратегій мінімізації відповідних ризиків. Серед потенційних

проблемних аспектів доцільно виокремити людський фактор у процесі введення даних, що може проявлятися у помилках користувачів при заповненні електронних форм замовлення, випадковому або навмисному внесенні некоректної інформації та недостатньому рівні навчання персоналу щодо використання системи. Обмеження мережевої інфраструктури включають залежність функціональності системи від стабільності інтернет-з'єднання, особливо у віддалених виробничих підрозділах або офісах, а також потенційні ризики перебоїв у роботі через технічні збої у мережевій інфраструктурі. Виклики міграційного періоду та організаційний спротив проявляються у необхідності трансферу історичних даних із існуючих Excel-файлів до нової системи, психологічному спротиві персоналу до нововведень («ми звикли до таблиць») та періоді адаптації, що може супроводжуватися тимчасовим зниженням продуктивності.

Питання технічної підтримки та оновлення системи актуалізуються через необхідність регулярного оновлення програмного забезпечення, усунення виявлених помилок, розширення функціоналу та ризики виникнення технічних проблем при відсутності належної системи підтримки або плану розвитку.

Недостатня диференціація прав доступу може генерувати потенційні ризики виникнення помилок при нечіткому розмежуванні прав між функціональними ролями та можливість несанкціонованої модифікації даних (наприклад, зміна технічних параметрів бухгалтером).

2.1.5 Поточний підхід до обробки замовлень (без інформаційної системи)

Аналіз поточної практики управління замовленнями на металообробних підприємствах демонструє превалювання табличного підходу на основі Microsoft Excel. Відповідно до цієї моделі, вся інформація про замовлення – від первинного запиту клієнта до фінального фінансового прорахунку –

акумулюється в єдиному файлі електронної таблиці. Процес ініціюється менеджером, який створює новий документ та заповнює відповідні поля, після чого файл послідовно передається іншим функціональним спеціалістам (технологу, бухгалтеру) для доповнення специфічною інформацією. Всі учасники процесу працюють із єдиним файлом у послідовному режимі, що створює низку системних обмежень та ризиків.

2.1.6 Проблемні місця в управлінні замовленнями при використанні Excel-таблиць

Незважаючи на визначеність функціональних ролей та послідовності процесів, управління замовленнями на основі Excel-таблиць характеризується наявністю низки системних обмежень, що негативно впливають на ефективність виробничих процесів. Відсутність централізованого середовища для взаємодії призводить до ситуації, коли дані не консолідовані у єдиній системі, що ускладнює доступ, контроль та аналіз інформації. Високий рівень ручної роботи проявляється у тому, що більшість розрахунків та введення даних виконуються в ручному режимі, що підвищує ризик виникнення помилок та знижує оперативність обробки замовлень.

Повільна трансмісія інформації між функціональними підрозділами виникає через необхідність ручної передачі файлів від менеджера технологу, а потім бухгалтеру, що генерує часові затримки та ризики втрати даних у процесі передачі.

Низька транспарентність статусів замовлень проявляється через відсутність єдиної точки моніторингу, що дозволила б відстежувати поточний етап обробки кожного замовлення та прогнозувати терміни виконання.

Складність аналітичної обробки та пошуку інформації зумовлена обмеженістю функцій фільтрації, сортування та аналізу даних, відсутністю контролю версій та аудиту змін, що ускладнює прийняття управлінських рішень на основі історичних даних.

Високий вплив людського фактору проявляється у ризиках втрати файлів, неузгоджених модифікацій, помилок у формулах розрахунків, що може суттєво вплинути на точність калькуляції та якість обслуговування клієнтів.

2.1.7 Переваги та недоліки використання Excel для управління замовленнями

Компаративний аналіз використання Excel-таблиць для управління замовленнями дозволяє ідентифікувати як певні переваги, так і суттєві обмеження даного підходу. Серед позитивних аспектів використання Excel доцільно виокремити інтуїтивність інтерфейсу, яка проявляється у відсутності необхідності спеціалізованого навчання персоналу, оскільки більшість співробітників вже знайомі з базовим функціоналом Excel. Гнучкість редагування надає можливість оперативного внесення будь-яких змін у структуру таблиці відповідно до змінних потреб та специфіки конкретного замовлення. Доступність Excel є його важливою перевагою, оскільки він є стандартним компонентом пакету Microsoft Office, встановленого на більшості корпоративних комп'ютерів, що не вимагає додаткових інвестицій у програмне забезпечення.

Водночас, Excel характеризується низкою критичних обмежень у контексті управління виробничими замовленнями. Відсутність повноцінного режиму багатокористувацького доступу проявляється у тому, що одночасна робота кількох спеціалістів із одним файлом є технічно ускладненою та створює ризики конфліктів при редагуванні. Високий ризик втрати або пошкодження файлу, особливо при передачі через електронну пошту або месенджери, може призвести до незворотної втрати даних.

Низька масштабованість проявляється у тому, що з ростом обсягів замовлень суттєво зростає складність структурування та аналізу інформації, що знижує ефективність управління при збільшенні масштабів діяльності.

Відсутність автоматизації змушує виконувати всі операції з розрахунку, фільтрації, аналізу та передачі інформації в ручному режимі, що підвищує трудомісткість процесів та знижує оперативність обробки замовлень. Складність контролю версій проявляється у відсутності механізмів відстеження, хто і коли вносив зміни у файл, що ускладнює аудит історії модифікацій та ідентифікацію відповідальних осіб.

Обмежений рівень безпеки означає, що доступ до файлу може отримати будь-який користувач, якому він був переданий, що створює ризики несанкціонованого доступу та модифікації даних.

Для узагальненої візуалізації інформаційних потоків у системі управління замовленнями розроблено контекстну діаграму потоків даних (DFD-0), представлену на рисунку 2.2, яка відображає взаємодію веб-системи із зовнішніми суб'єктами, включаючи клієнтів, користувачів та зовнішні сервіси.

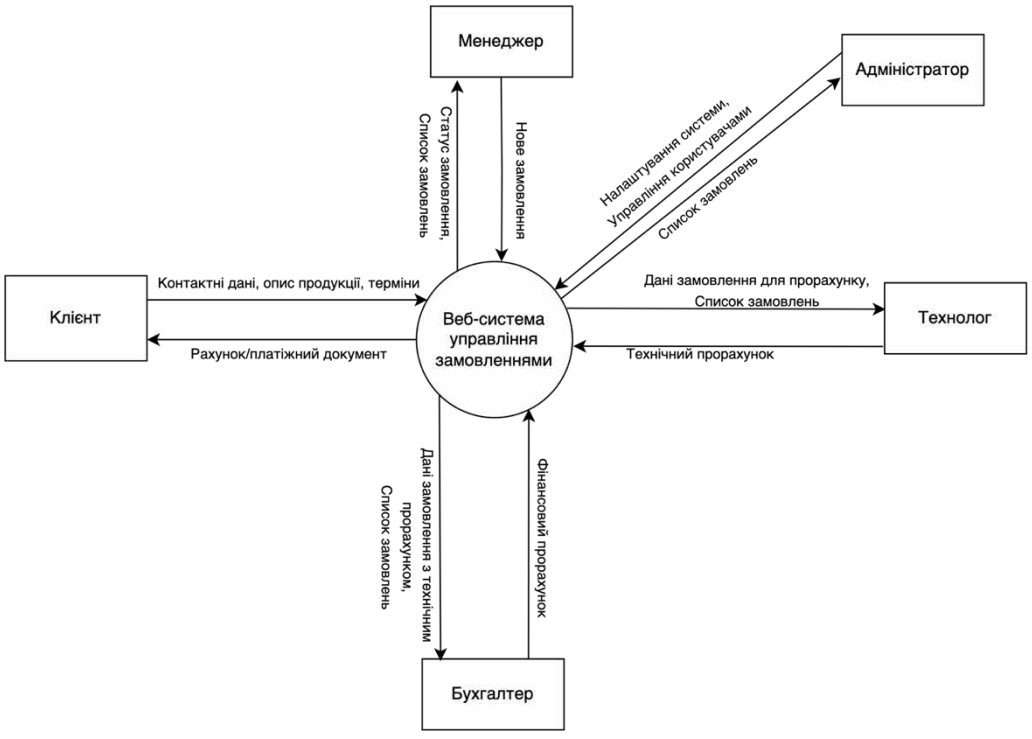


Рисунок 2.2 – Контекстна діаграма потоків даних веб-системи управління замовленнями

На діаграмі візуалізовано процес взаємодії клієнта з системою, де клієнт надає первинні дані для замовлення, які вносяться у систему менеджером. Система забезпечує комплексну обробку цих даних, передаючи їх послідовно технологу та бухгалтеру відповідно до визначеного процесуального алгоритму, та генерує вихідні документи, включаючи рахунки, підтвердження замовлення та аналітичні звіти. Структуризація інформаційних потоків у рамках єдиної системи забезпечує підвищення ефективності управління замовленнями та мінімізацію ризиків втрати або спотворення даних.

2.2 Визначення архітектури системи

2.2.1 Клієнт-серверна архітектура системи та взаємодія компонентів

Архітектурна концепція веб-системи управління замовленнями базується на класичній клієнт-серверній моделі, яка є домінуючою парадигмою для сучасних веб-застосунків. Відповідно до цієї моделі, клієнтська частина (фронтенд) взаємодіє із серверною частиною (бекенд) через стандартизований протокол HTTP/HTTPS. Така архітектурна парадигма забезпечує чітке розмежування відповідальності між презентаційним рівнем (інтерфейс користувача) та логічним рівнем (обробка даних, автентифікація, доступ до бази даних), що підвищує модульність, масштабованість та надійність системи.

За типологічною класифікацією, розроблена система належить до категорії багаторівневих (multi-tier) архітектур, що характеризуються чітким структурним розподілом на функціональні рівні з визначеними інтерфейсами взаємодії. Архітектурна декомпозиція системи включає три ключові рівні.

Презентаційний рівень (Frontend / Angular) забезпечує реалізацію користувацького інтерфейсу та взаємодію з кінцевим користувачем. Він відповідає за візуалізацію даних, обробку користувацьких дій та формування

запитів до серверної частини. Технологічно цей рівень базується на фреймворку Angular з використанням TypeScript, HTML, SCSS та супутніх бібліотек (RxJS, NgRx) для забезпечення реактивності інтерфейсу, управління станом та асинхронної обробки запитів. Взаємодія з бекендом реалізується через інтерфейс REST API.

Рівень бізнес-логіки (Backend / Java + Spring) реалізує основну бізнес-логіку системи, включаючи обробку запитів від клієнтської частини, валідацію даних, автентифікацію та авторизацію користувачів, логіку розрахунків та перевірок. Технологічно цей рівень імплементований на мові Java з використанням фреймворку Spring Boot, що забезпечує високий рівень модульності, надійності та продуктивності. Він інтегрує механізми безпеки, зокрема OAuth для управління автентифікацією та авторизацією, та забезпечує взаємодію з рівнем даних для зберігання та отримання інформації.

Рівень даних (Database / MongoDB) відповідає за зберігання, організацію та управління даними системи. Технологічно цей рівень базується на документно-орієнтованій NoSQL базі даних MongoDB, що забезпечує оптимальну продуктивність при роботі з складноструктурованими об'єктами (замовлення, матеріали, користувачі). Інтеграція з рівнем бізнес-логіки реалізована через Spring Data MongoDB, що забезпечує зручний інтерфейс взаємодії між Java-сервісами та колекціями MongoDB.

Схематична візуалізація архітектурної моделі системи представлена на рисунку 2.3, де відображено взаємозв'язки між ключовими компонентами та технологічними рішеннями.

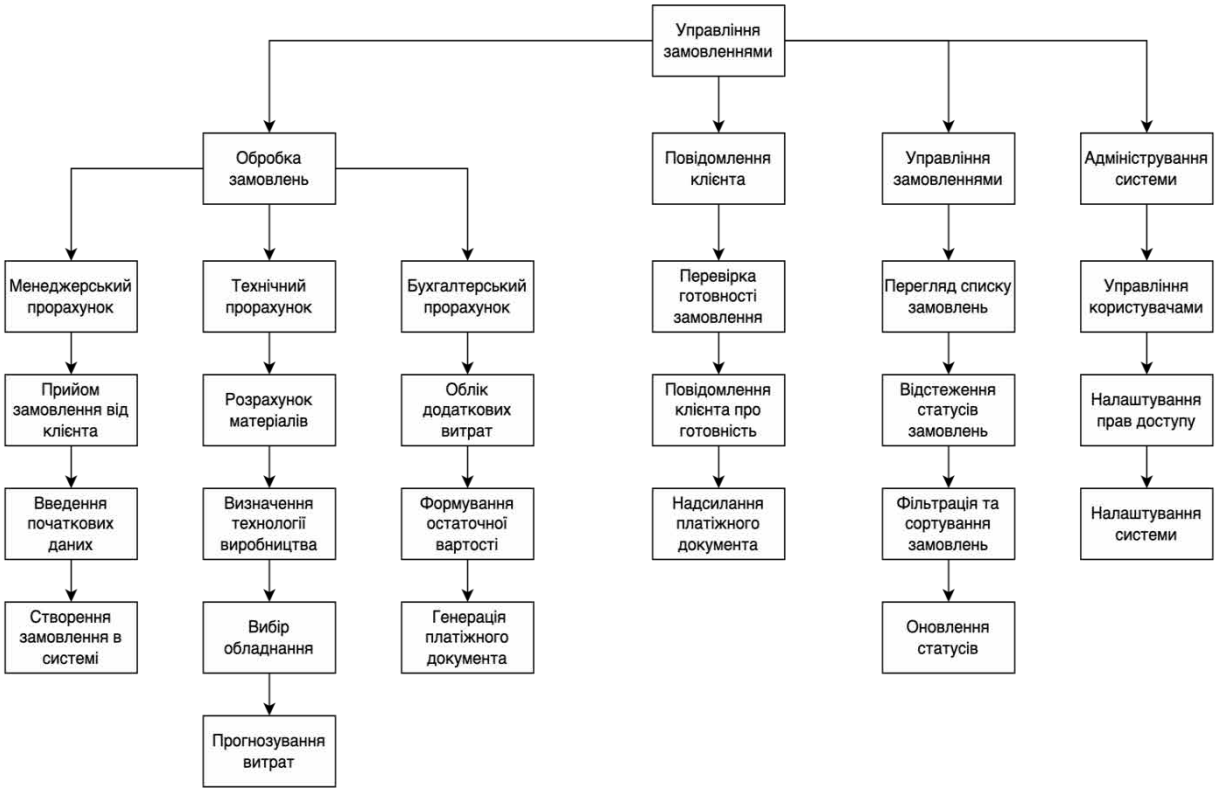


Рисунок 2.3 – Архітектурна модель веб-системи управління замовленнями

2.2.2 Технологічні основи та вибір архітектури

Визначення оптимального технологічного стеку для розробки веб-системи управління замовленнями базувалося на комплексному аналізі функціональних вимог, технічних обмежень та стратегічних цілей проєкту. Ключовими факторами, що впливали на вибір технологій, були специфіка предметної області та типовий обсяг даних, що проявляється у необхідності ефективної обробки складноструктурованих даних про замовлення з різними статусами, категоріями та ієрархічними зв'язками.

Вимоги до інтеграції з різними сервісами та API визначають потребу в гнучких механізмах взаємодії з зовнішніми системами, включаючи платіжні шлюзи, бухгалтерські програми, системи CRM.

Масштабованість та підтримка системи є важливими факторами, що проявляються у необхідності забезпечення можливості розширення

функціоналу та адаптації до зростаючих обсягів даних без суттєвої перебудови архітектури.

Наявність активної спільноти та документації визначає важливість використання технологій з розвиненою екосистемою, що забезпечує доступ до навчальних матеріалів, бібліотек, інструментів та експертної підтримки.

Продуктивність та безпека є критичними факторами, що проявляються у необхідності забезпечення високої швидкодії системи та надійного захисту даних, особливо в контексті роботи з комерційною інформацією.

Для реалізації клієнтської частини веб-системи обрано фреймворк Angular – потужне рішення з відкритим кодом для створення односторінкових застосунків (Single Page Applications, SPA). Angular розроблений та підтримується компанією Google, що забезпечує високий рівень надійності та довгострокової підтримки. Технологічний фундамент Angular базується на мові TypeScript – розширенні JavaScript, що додає статичну типізацію, класи, інтерфейси та інші компоненти об'єктно-орієнтованої парадигми.

На відміну від бібліотек для побудови користувацького інтерфейсу, таких як React, Angular представляє собою повноцінний фреймворк, що охоплює всі аспекти розробки клієнтської частини: маршрутизацію, управління формами, обробку HTTP-запитів, управління станом, анімації, модульність та тестування. Це забезпечує комплексний та уніфікований підхід до розробки, що підвищує консистентність коду та спрощує його підтримку.

Ключові аргументи на користь вибору Angular включають компонентну архітектуру, яка базується на використанні ізольованих, повторно використовуваних компонентів, що відповідають за окремі елементи інтерфейсу. Така модульність спрощує розробку, тестування та підтримку складних інтерфейсів, характерних для систем управління бізнес-процесами.

Статична типізація через TypeScript забезпечує раннє виявлення помилок на етапі компіляції, покращує читабельність коду та підвищує загальну надійність системи, що особливо важливо для бізнес-критичних застосунків.

Інструментарій командного рядка (Angular CLI) надає офіційний CLI-інструмент для генерації проєктів, компонентів, сервісів, модулів та запуску тестів, що суттєво прискорює процес розробки та забезпечує стандартизацію структури коду.

Декларативні HTML-шаблони з двостороннім зв'язуванням даних у Angular підтримують розширену HTML-синтаксис для створення інтерактивних інтерфейсів через механізми прив'язки змінних, обробки подій, умовного відображення та циклів, що мінімізує обсяг JavaScript-коду та підвищує читабельність шаблонів. Вбудована система маршрутизації (Angular Router) забезпечує реалізацію SPA-архітектури з гнучкою навігацією, ледачим завантаженням модулів (lazy loading), параметризованими маршрутами та вкладеними навігаційними структурами, що критично для побудови складних багатосторінкових інтерфейсів. Захисні механізми маршрутів (Guards) надають інструменти для обмеження доступу до певних маршрутів залежно від статусу автентифікації або ролі користувача, що є необхідним для системи з диференційованим доступом (менеджер, технолог, бухгалтер, адміністратор).

Модульність та механізми ледачого завантаження забезпечують можливість розділення проєкту на окремі функціональні модулі з динамічним завантаженням за потреби, що знижує початкове навантаження на браузер та оптимізує продуктивність системи. Для розширення функціональних можливостей Angular та оптимізації розробки використано додаткові технології, серед яких TypeScript – типізована мова програмування, що компілюється у JavaScript та забезпечує надійність коду, спрощує рефакторинг та виявлення помилок на ранніх етапах, що критично для масштабних проєктів з складною бізнес-логікою.

RxJS є бібліотекою для реактивного програмування, що є інтегральною частиною Angular та забезпечує ефективну роботу з асинхронними потоками даних, включаючи HTTP-запити, події користувацького інтерфейсу, таймери

та реакції на зміни стану. RxJS дозволяє імплементувати складні сценарії асинхронної взаємодії без надмірного ускладнення імперативного коду.

NgRx є бібліотекою для централізованого управління станом на основі патерну Redux, що забезпечує єдине джерело істини для даних, уніфікацію потоків інформації, механізми кешування та обробку асинхронних подій. Це особливо важливо для системи з великими обсягами даних, які мають синхронізуватися між різними компонентами інтерфейсу (наприклад, статуси замовлень, профілі користувачів, історія дій).

SCSS є препроцесором CSS, що розширює стандартний синтаксис таблиць стилів можливостями використання змінних, міксинів, вкладених селекторів, умовних операторів та циклів. Це підвищує модульність, читабельність та повторне використання стилів, що важливо для забезпечення візуальної консистентності великих інтерфейсів.

REST API є комунікаційним протоколом для взаємодії з серверною частиною, що базується на стандартних HTTP-методах та форматі JSON. Angular має вбудований HTTP-клієнт, що забезпечує зручний інтерфейс для роботи з REST-сервісами, включаючи обробку запитів, відповідей, помилок, заголовків та автентифікаційних токенів.

Для реалізації серверної частини веб-системи обрано технологічний стек на основі мови Java та фреймворку Spring Boot, що забезпечує надійну платформу для розробки корпоративних застосунків з високими вимогами до продуктивності, безпеки та масштабованості.

Java є однією з найпопулярніших мов програмування для розробки серверних застосунків, що характеризується крос-платформеністю, потужною стандартною бібліотекою, строгою типізацією та розвиненою екосистемою фреймворків і бібліотек. Використання Java забезпечує надійність, безпеку та продуктивність серверної частини, що критично для бізнес-орієнтованих систем. Spring Boot є фреймворком, що спрощує розробку застосунків на основі Spring через механізми автоконфігурації, вбудований веб-сервер та мінімізацію ручного налаштування. Spring Boot забезпечує швидкий старт

розробки, інтеграцію з різними технологіями та високу продуктивність, що робить його оптимальним вибором для створення REST API та бізнес-логіки системи управління замовленнями. Ключові переваги використання Java та Spring Boot для бекенду включають модульну архітектуру, яка базується на принципах інверсії контролю (IoC) та впровадження залежностей (DI), що забезпечує високий рівень модульності, тестованості та підтримки коду.

Вбудована підтримка REST через Spring MVC та Spring Web забезпечує зручний інструментарій для створення RESTful API з мінімальним обсягом шаблонного коду та високим рівнем абстракції. Безпека через Spring Security надає комплексні механізми автентифікації, авторизації та захисту від поширених вразливостей веб-застосунків, включаючи підтримку OAuth 2.0 та JWT-токенів.

Інтеграція з базами даних через Spring Data забезпечує уніфікований інтерфейс для роботи з різними типами баз даних, включаючи MongoDB, що спрощує реалізацію операцій створення, читання, оновлення та видалення даних.

Аспектно-орієнтоване програмування (AOP) у Spring підтримує AOP, що дозволяє реалізувати наскрізну функціональність (логування, безпека, транзакції) без дублювання коду та змішування різних аспектів бізнес-логіки. Масштабованість Spring Boot підтримує горизонтальне масштабування через мікросервісну архітектуру, що дозволяє системі адаптуватися до зростаючого навантаження та обсягу даних. Управління транзакціями у Spring забезпечує декларативне управління транзакціями, що спрощує забезпечення цілісності даних при складних операціях, характерних для систем управління бізнес-процесами.

Для зберігання та управління даними веб-системи обрано MongoDB – документно-орієнтовану NoSQL базу даних, що забезпечує високу продуктивність, гнучкість схеми та горизонтальну масштабованість. На відміну від традиційних реляційних баз даних, MongoDB зберігає дані у форматі JSON-подібних документів, що особливо ефективно для роботи з

складноструктурованими об'єктами, характерними для системи управління замовленнями.

Ключові переваги використання MongoDB включають гнучкість схеми, яка не вимагає визначення фіксованої структури даних перед їх збереженням, що дозволяє легко адаптуватися до змінних вимог та еволюції моделі даних без складних міграцій схеми.

Документно-орієнтована модель забезпечує зберігання даних у вигляді документів, що відповідають об'єктам у коді, що спрощує маппінг між об'єктною моделлю застосунку та структурою зберігання, мінімізуючи об'єктно-реляційний імпеданс.

Висока продуктивність MongoDB досягається через оптимізацію для операцій читання та запису, що критично для систем з інтенсивним доступом до даних, характерних для управління виробничими процесами.

Горизонтальна масштабованість забезпечується через вбудовану підтримку шардингу (розподілу даних між серверами), що дозволяє системі ефективно масштабуватися при зростанні обсягу даних та кількості користувачів.

Агрегаційний конвеєр MongoDB є потужним механізмом для аналізу та обробки даних безпосередньо в базі даних, що оптимізує генерацію звітів та аналітичних виборок, характерних для бізнес-систем.

Інтеграція з Spring реалізується через бібліотеку Spring Data MongoDB, що забезпечує зручний інтерфейс для взаємодії з MongoDB з використанням об'єктно-орієнтованої парадигми, що спрощує розробку та підтримку коду.

Інтеграція MongoDB з Java-сервісами реалізована через Spring Data MongoDB, що надає високорівневий абстрактний інтерфейс для роботи з документами, колекціями та запитами.

2.3 Комунікація між фронтендом і бекендом через REST API

Архітектурною основою взаємодії між клієнтською (фронтенд) та серверною (бекенд) частинами веб-системи управління замовленнями виступає стиль REST (Representational State Transfer), що базується на стандартизованому протоколі HTTP/HTTPS. REST API є фундаментальним механізмом обміну даними між Angular-фронтендом та Java Spring Boot-бекендом, забезпечуючи модульність, масштабованість та гнучкість системної архітектури. Даний підхід оптимізує обробку користувацьких запитів, трансмісію даних між різними рівнями архітектури та інтеграцію з зовнішніми системами, такими як платіжні шлюзи або бухгалтерські програми.

2.3.1 Принципи роботи REST API в системі

REST API функціонує за класичною клієнт-серверною парадигмою, де фронтенд виступає клієнтом, що генерує запити до бекенду, а бекенд забезпечує обробку цих запитів та формування відповідей. Комунікація реалізується через стандартизовані HTTP-методи, такі як GET, POST, PUT, DELETE, що відповідають базовим операціям CRUD (Create, Read, Update, Delete). Кожен запит та відповідь форматуються у JSON (JavaScript Object Notation), що забезпечує оптимальну сумісність та зручність обробки даних як на клієнтській, так і на серверній стороні.

Базові принципи REST, імплементовані в системі, включають статичність (Stateless), яка проявляється у тому, що кожен запит від фронтенду до бекенду є самодостатнім та містить всю необхідну інформацію для його обробки. Бекенд не зберігає сесійного стану між запитами, що спрощує масштабування системи та підвищує її відмовостійкість.

Ресурсна орієнтованість означає, що всі об'єкти системи (замовлення, користувачі, матеріали) представлені як ресурси, доступ до яких здійснюється через уніфіковані URL-адреси (ендпоінти), наприклад, /api/orders для роботи із замовленнями. Стандартизація HTTP-методів проявляється у використанні методів HTTP для чіткого визначення типу операції. Наприклад, POST

/api/orders створює нове замовлення, GET /api/orders/{id} отримує дані конкретного замовлення, PUT /api/orders/{id} оновлює існуюче замовлення, а DELETE /api/orders/{id} видаляє замовлення. Кешування як принцип REST означає, що для оптимізації продуктивності відповіді на запити GET можуть кешуватися на клієнтській стороні, що мінімізує навантаження на сервер та прискорює відображення даних. Багаторівневість REST API проявляється у підтримці інтеграції проміжних компонентів (балансувальники навантаження, проксі-сервери), що сприяє масштабованості та надійності системи.

2.3.2 Механізм обробки запитів

Процесуальний алгоритм комунікації між фронтендом та бекендом через REST API структурується за трьома основними етапами.

Формування запиту на фронтенді починається з того, що Angular-фронтенд використовує вбудований модуль HttpClient для створення HTTP-запитів. Наприклад, при створенні нового замовлення менеджер заповнює електронну форму, після чого Angular формує JSON-об'єкт з даними замовлення (контактні дані клієнта, специфікація продукції, терміни виконання) та надсилає його у запиті POST /api/orders. Запити включають службові заголовки (headers), зокрема Content-Type: application/json для визначення формату даних, а також Authorization з токеном автентифікації для ідентифікації користувача.

Обробка запиту на бекенді відбувається так, що Java Spring Boot-бекенд отримує запит через REST-контролери (анотовані як @RestController). Наприклад, контролер OrderController обробляє запит POST /api/orders, викликаючи відповідний сервіс для валідації даних, збереження замовлення в MongoDB та генерації відповіді. Spring Security забезпечує автентифікацію та авторизацію, верифікуючи наявність у користувача (наприклад, менеджера) відповідних прав на виконання запитуваної операції. Після обробки даних бекенд формує відповідь у форматі JSON, що містить статус операції

(наприклад, код 201 Created для успішного створення замовлення) та, за необхідності, додаткові дані (наприклад, ідентифікатор створеного замовлення).

Обробка відповіді на фронтенді полягає у тому, що Angular отримує відповідь та обробляє її з використанням RxJS-операторів (map, catchError) для відповідного оновлення інтерфейсу користувача. Наприклад, після успішного створення замовлення система відображає повідомлення про успіх та перенаправляє менеджера до списку замовлень. У випадку виникнення помилки (наприклад, код 400 Bad Request через невалідні дані) фронтенд відображає відповідне повідомлення, як показано на рисунку 2.10, що демонструє обробку помилок валідації.

2.3.3 Безпека комунікації

Для забезпечення безпеки інформаційного обміну між фронтендом та бекендом імплементовано комплекс захисних механізмів. HTTPS забезпечує передачу всіх запитів через захищений протокол HTTPS, що гарантує шифрування даних під час передачі та запобігає їх перехопленню або модифікації.

Автентифікація через OAuth 2.0 полягає у тому, що кожен запит супроводжується токеном доступу (JWT – JSON Web Token), що генерується після успішної автентифікації користувача. Spring Security верифікує валідність токену, забезпечуючи доступ лише авторизованим користувачам.

Рольовий доступ реалізується через Spring Boot, який використовує конфігурацію Spring Security для обмеження доступу до ендпоінтів відповідно до ролі користувача. Наприклад, лише бухгалтер може викликати POST /api/financials, а лише адміністратор – DELETE /api/users/{id}.

Валідація даних здійснюється на бекенді з використанням бібліотеки Hibernate Validator для верифікації вхідних даних, що запобігає обробці некоректних або потенційно шкідливих запитів.

2.3.4 Переваги використання REST API

Імплементація REST API у веб-системі управління замовленнями забезпечує низку суттєвих переваг. Гнучкість проявляється у тому, що REST API дозволяє легко розширювати функціональність через додавання нових ендпоінтів або модифікацію існуючих без необхідності зміни клієнтської частини, що спрощує еволюцію системи. Масштабованість забезпечується завдяки статичності запитів та механізмам кешування, що дозволяє системі ефективно обробляти велику кількість одночасних запитів, що критично для підприємств з високим обсягом замовлень. Інтеграційний потенціал REST API проявляється у забезпеченні стандартизованого інтерфейсу для інтеграції з зовнішніми системами, такими як CRM, платіжні шлюзи або бухгалтерські програми, через уніфіковані HTTP-запити. Кросплатформеність REST API дозволяє замінити фронтенд на альтернативне рішення (наприклад, мобільний додаток) без необхідності модифікації бекенду, оскільки REST API є універсальним інтерфейсом для обміну даними. Тестованість REST API проявляється у тому, що ендпоінти легко тестуються з використанням спеціалізованих інструментів, таких як Postman або Swagger, що прискорює розробку та відлагодження.

2.3.5 Обмеження та шляхи їх подолання

Незважаючи на численні переваги, використання REST API супроводжується певними обмеженнями, для яких розроблено відповідні компенсаторні механізми. Залежність від мережевої інфраструктури означає, що для стабільної роботи системи необхідне надійне інтернет-з'єднання. Для мінімізації цього обмеження імплементовано механізми кешування даних на клієнтській стороні та періодичної синхронізації при відновленні з'єднання.

Обсяг даних у запитах може бути проблемою, оскільки великі обсяги даних у JSON-відповідях можуть негативно впливати на швидкість обробки. Для оптимізації використано пагінацію (GET /api/orders?page=1&size=10) та компресію даних (gzip).

Складність управління токенами полягає у тому, що токени JWT мають обмежений термін дії, що вимагає періодичного оновлення. Для вирішення цієї проблеми імплементовано механізм refresh-токенів, що забезпечує автоматичне оновлення токену доступу без необхідності повторної автентифікації користувача.

2.3.6 Опис методу getOrders

Метод getOrders є ключовим компонентом сервісного шару системи, що відповідає за взаємодію з REST API для отримання та обробки даних про замовлення. Основною функціональністю методу є отримання пагінованого списку замовлень з бекенду з урахуванням параметрів фільтрації, сортування, пошуку та вибраної користувацької вкладки, що визначає статуси замовлень для відображення.

Даний метод використовується для формування таблиць замовлень у веб-інтерфейсі, зокрема таблиці «Замовлення» (статуси: PAID, NON_PAID, SET_OFF) для відображення активних замовлень та таблиці «Нові замовлення» (статуси: WAITING, COUNTING, SENDING) для моніторингу замовлень на етапі обробки.

Метод повертає об'єкт типу Observable<Page<OrderResponse>>, що забезпечує реактивну обробку відповіді від сервера, динамічне оновлення інтерфейсу користувача та ефективну обробку потенційних помилок з використанням функціоналу RxJS.

Параметри методу включають page: number = DEFAULT_PAGE_NUMBER, що є номером сторінки для пагінації (нумерація починається з 0), зі значенням за замовчуванням DEFAULT_PAGE_NUMBER

(визначено як константа в коді), та використовується для визначення, яку сторінку замовлень необхідно отримати.

Параметр size: number = DEFAULT_PAGE_SIZE визначає кількість записів (замовлень) на одній сторінці, зі значенням за замовчуванням DEFAULT_PAGE_SIZE (визначено як константа), та дозволяє контролювати обсяг даних, що отримуються в одному запиті.

Параметр sortType: SortType = SortType.DEFAULT визначає тип сортування замовлень (перерахування SortType з варіантами ASC для висхідного або DESC для низхідного порядку), зі значенням за замовчуванням SortType.DEFAULT (сортування за датою виконання), та визначає порядок сортування результатів за полем execution_date.

Параметр searchValue: string = '' є рядком для пошуку замовлень за назвою клієнта (підтримує частковий збіг), зі значенням за замовчуванням – порожній рядок (пошук не застосовується), та використовується для фільтрації замовлень за клієнтом.

Параметр tab: string = APP_ROUTES.ORDERS визначає вкладку інтерфейсу, що визначає, які статуси замовлень необхідно отримати, зі значенням за замовчуванням APP_ROUTES.ORDERS (константа, що відповідає вкладці «Замовлення в роботі»), та використовується для диференціації між вкладками «Замовлення в роботі» та «Нові замовлення».

Повернене значення методу – Observable<Page<OrderResponse>>, тобто об'єкт типу Observable, що містить сторінку замовлень (Page<OrderResponse>). Page є інтерфейсом, що містить список замовлень (OrderResponse[]) та метадані пагінації (загальна кількість сторінок, загальна кількість записів). OrderResponse є DTO (Data Transfer Object), що представляє замовлення з полями, такими як ідентифікатор, статус, клієнт, дата виконання тощо. Використання Observable забезпечує асинхронну обробку відповіді від сервера та реактивне оновлення інтерфейсу.

Таблиця 2.4 – Основні REST-запити системи

Метод	Ендпоінт	Опис	Параметри / Тіло запиту	Відповідальні ролі
GET	/order-manager/v1/orders	Отримати список замовлень	page, size, sort, query (фільтр за статусом, пошук по клієнту)	Менеджер, Адміністратор
GET	/order-manager/v1/orders/{number}	Отримати деталі одного замовлення	{number} – номер замовлення	Менеджер, Технолог, Бухгалтер
POST	/order-manager/v1/orders	Створити нове замовлення (менеджерський прорахунок)	JSON ManagerCalculationRequest	Менеджер
PATCH	/order-manager/v1/orders/{number}	Оновити існуючий менеджерський прорахунок	{number}, JSON ManagerCalculationRequest	Менеджер
GET	/specification-service/v1/orders/{orderNumber}/specifications	Отримати список специфікацій за номером замовлення	{orderNumber}, page, size	Технолог

2.4 Інтерфейс користувача та логіка обробки замовлень

Інтерфейс користувача веб-системи управління замовленнями розроблено з дотриманням принципів ергономічності, інтуїтивності та функціональної ефективності. Архітектурна концепція інтерфейсу базується на адаптивному дизайні, що забезпечує коректне відображення на різних типах пристроїв, та модульній структурі, що диференціює доступ до функціоналу відповідно до ролі користувача.

2.4.1 Сторінка авторизації

Початковою точкою взаємодії користувача з системою є сторінка авторизації, представлена на рисунку 2.4. Сторінка забезпечує автентифікацію користувачів через введення логіну та паролю, з подальшою валідацією облікових даних та перенаправленням до відповідного функціонального інтерфейсу залежно від ролі.

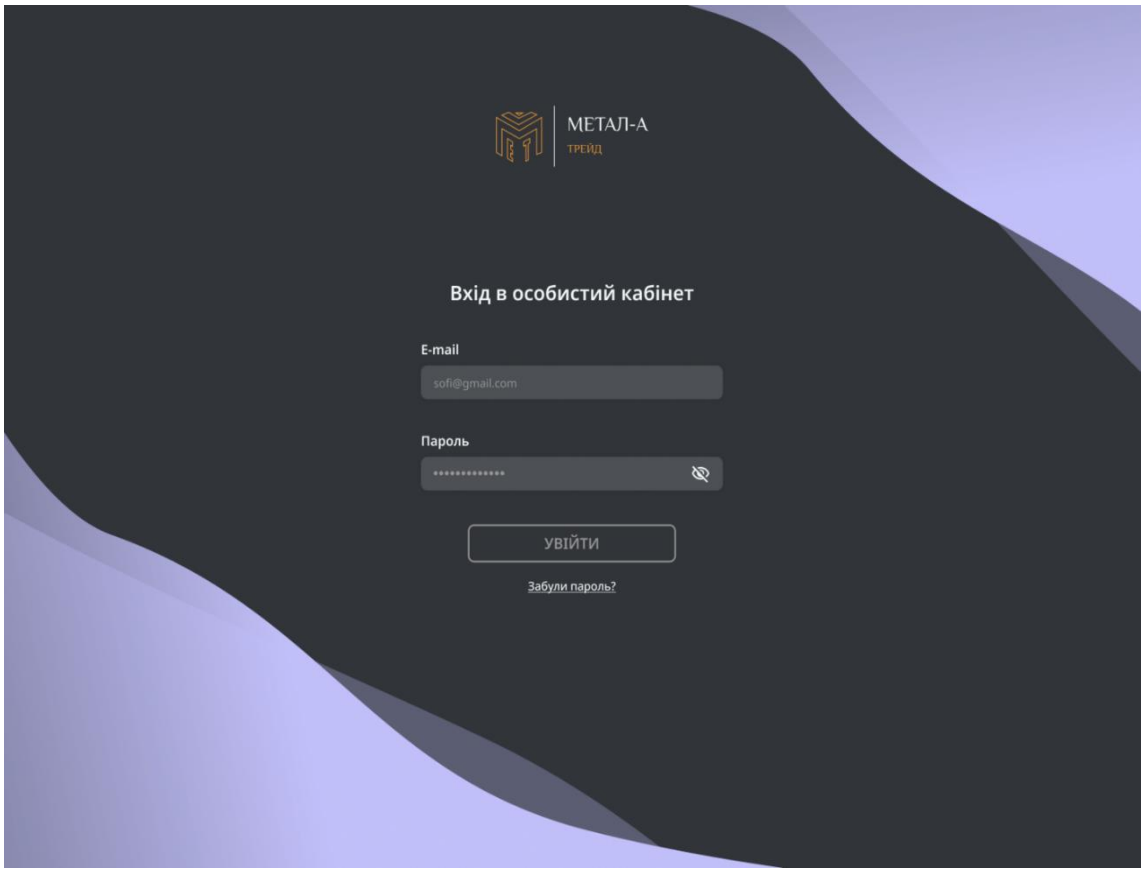


Рисунок 2.4 – Сторінка авторизації співробітника

Система реалізує механізми валідації введених даних з відображенням відповідних повідомлень про помилки у випадку некоректного заповнення полів або відсутності необхідних даних, як показано на рисунку 2.5. Це забезпечує інформативний зворотний зв'язок та покращує користувацький досвід при взаємодії з системою.

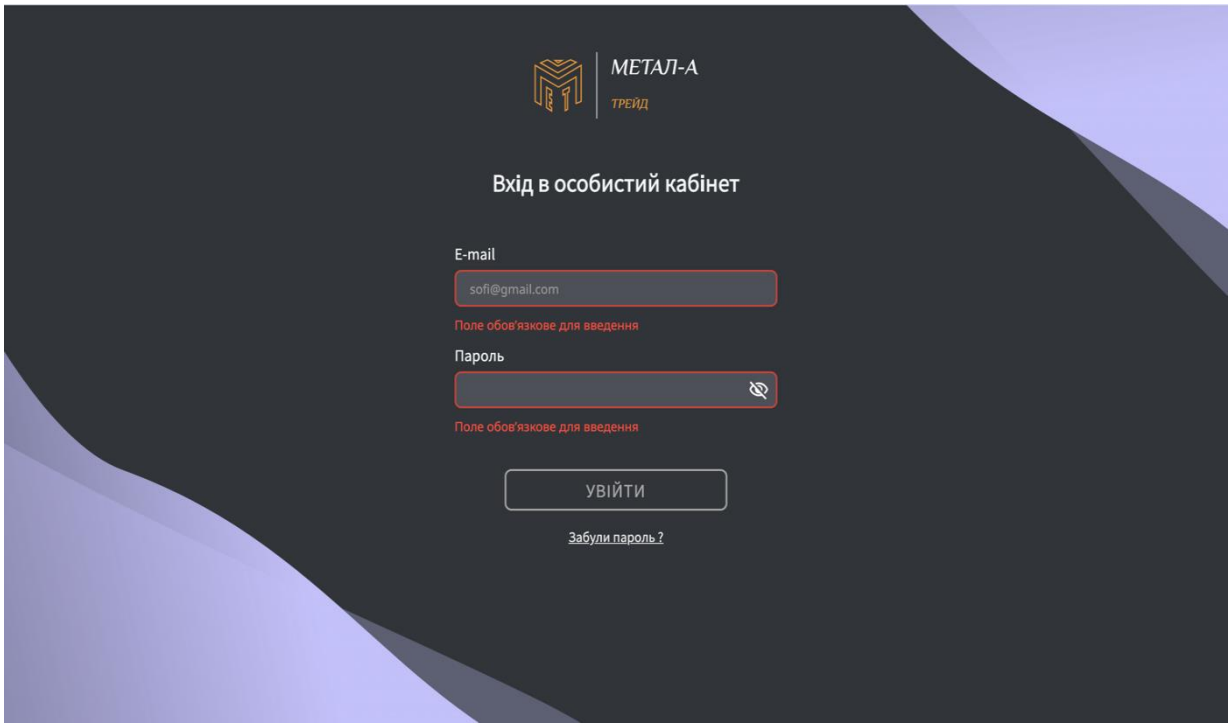


Рисунок 2.5 – Обробка помилок при некоректному заповненні даних та їх відсутності

2.4.2 Таблиця замовлень

Основним інструментом управління замовленнями є табличний інтерфейс, що відображає перелік замовлень з їх ключовими параметрами та статусами. Система реалізує два основні режими відображення замовлень.

Таблиця «Замовлення в роботі» (рисунок 2.6) відображає замовлення, що знаходяться на різних етапах обробки після завершення початкових прорахунків. Кожне замовлення має відповідний статус, пов'язаний з оплатою (PAID, NON_PAID, SET_OFF), та містить коротку інформацію про клієнта, дату, статус та інші ключові параметри.

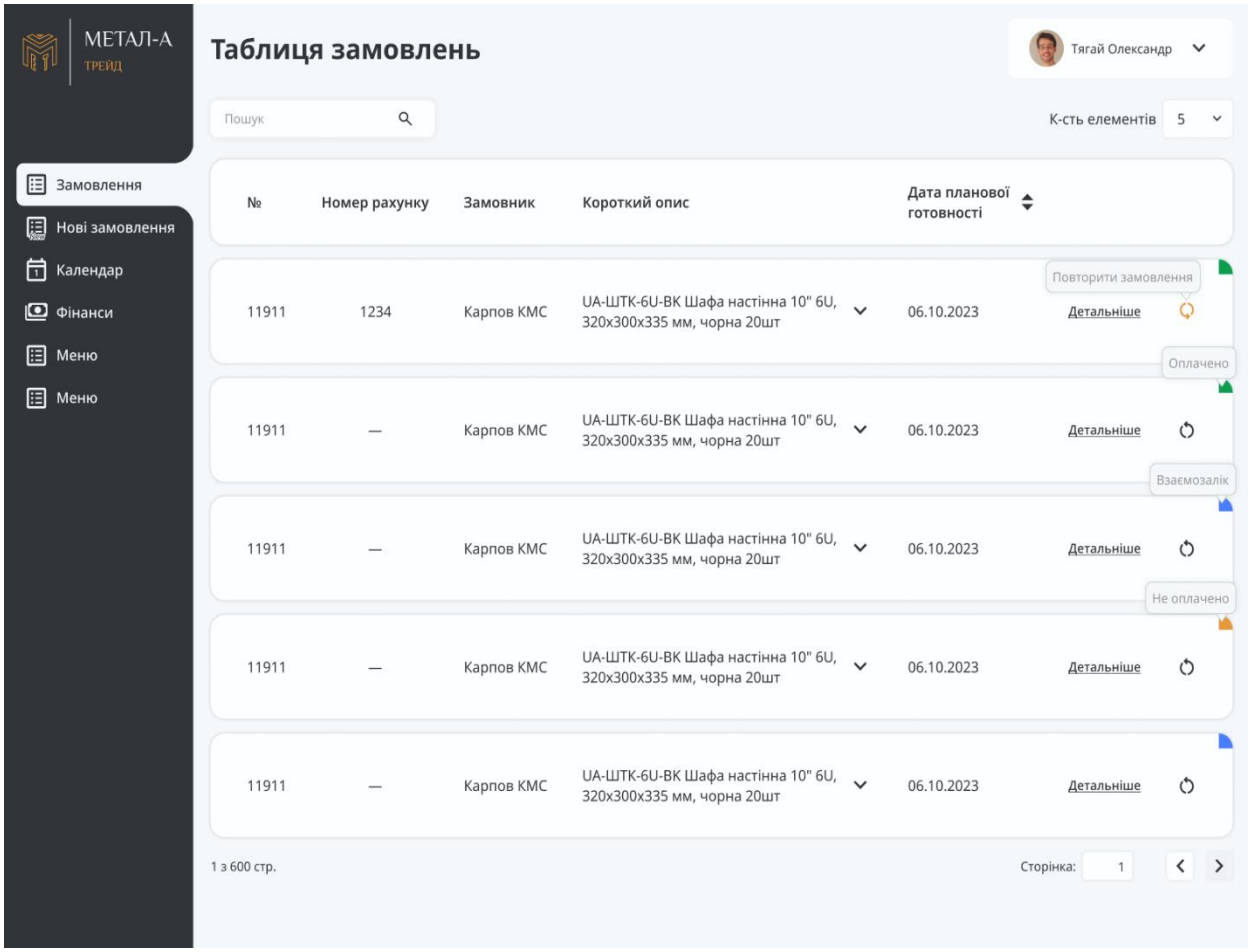


Рисунок 2.6 – Таблиця замовлень, які знаходяться в роботі

Таблиця «Нові замовлення» (Рис. 2.7) відображає замовлення, які щойно створені в системі та знаходяться на початкових етапах обробки (WAITING, COUNTING, SENDING). Таблиця забезпечує моніторинг статусу обробки та дозволяє оперативно реагувати на нові запити.

Обидві таблиці забезпечують інтерактивну функціональність, включаючи сортування, фільтрацію, пагінацію та пошук, що оптимізує процес управління великими обсягами замовлень.

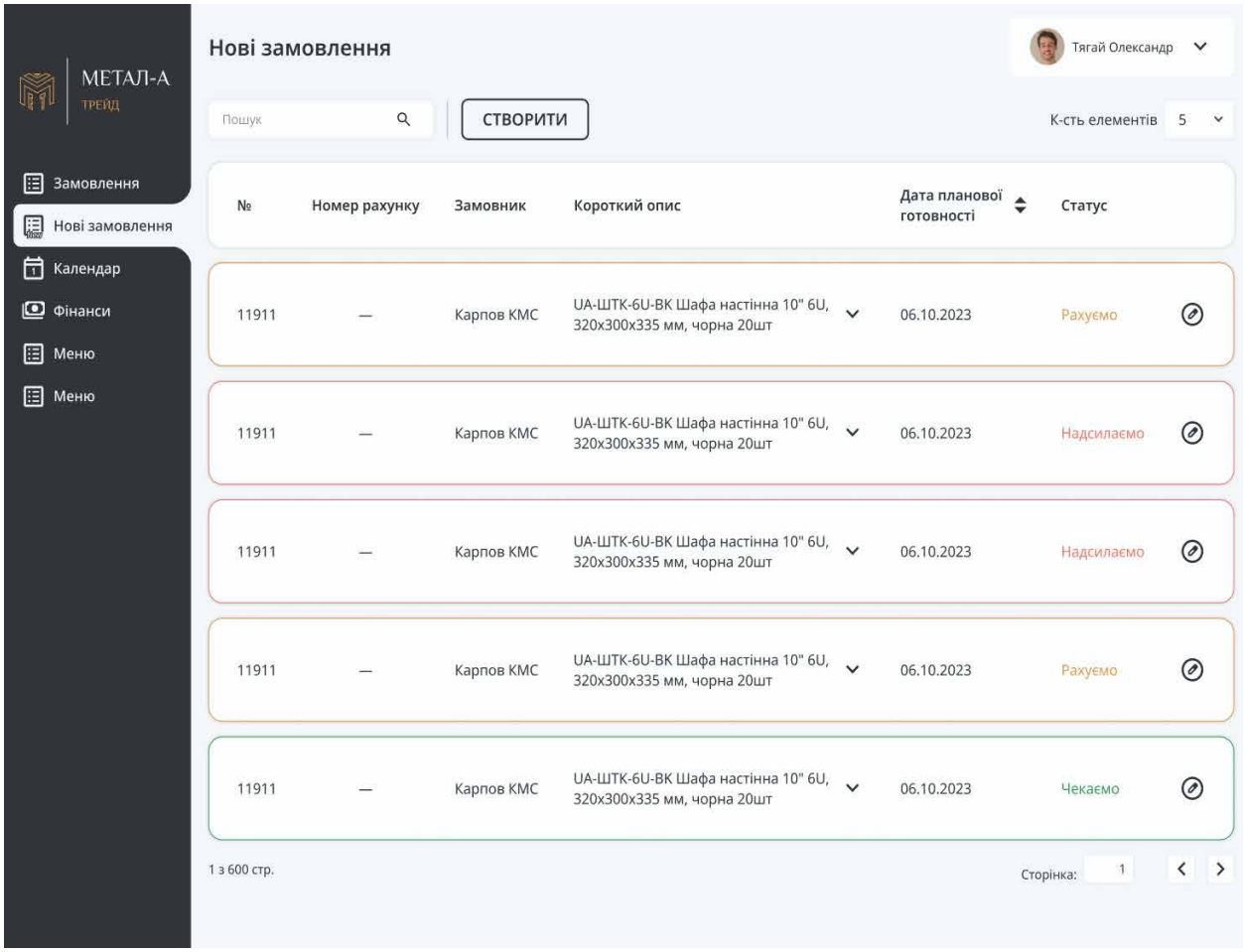


Рисунок 2.7 – Таблиця нових замовлень

2.4.3 Форми обробки замовлень

Система реалізує спеціалізовані форми для кожного етапу обробки замовлення, відповідно до функціональних ролей користувачів. Форма менеджерського прорахунку (Рис. 2.8) забезпечує введення базової інформації для початкового створення замовлення. Форма включає поля для заповнення контактних даних клієнта, специфікації продукції, термінів виконання та додаткових вимог. Для підвищення зручності використання реалізовано кастомні компоненти для вибору значень (dropdown) та багатоваріантного вибору (checkbox), що забезпечує більш гнучкий контроль над введенням даних.

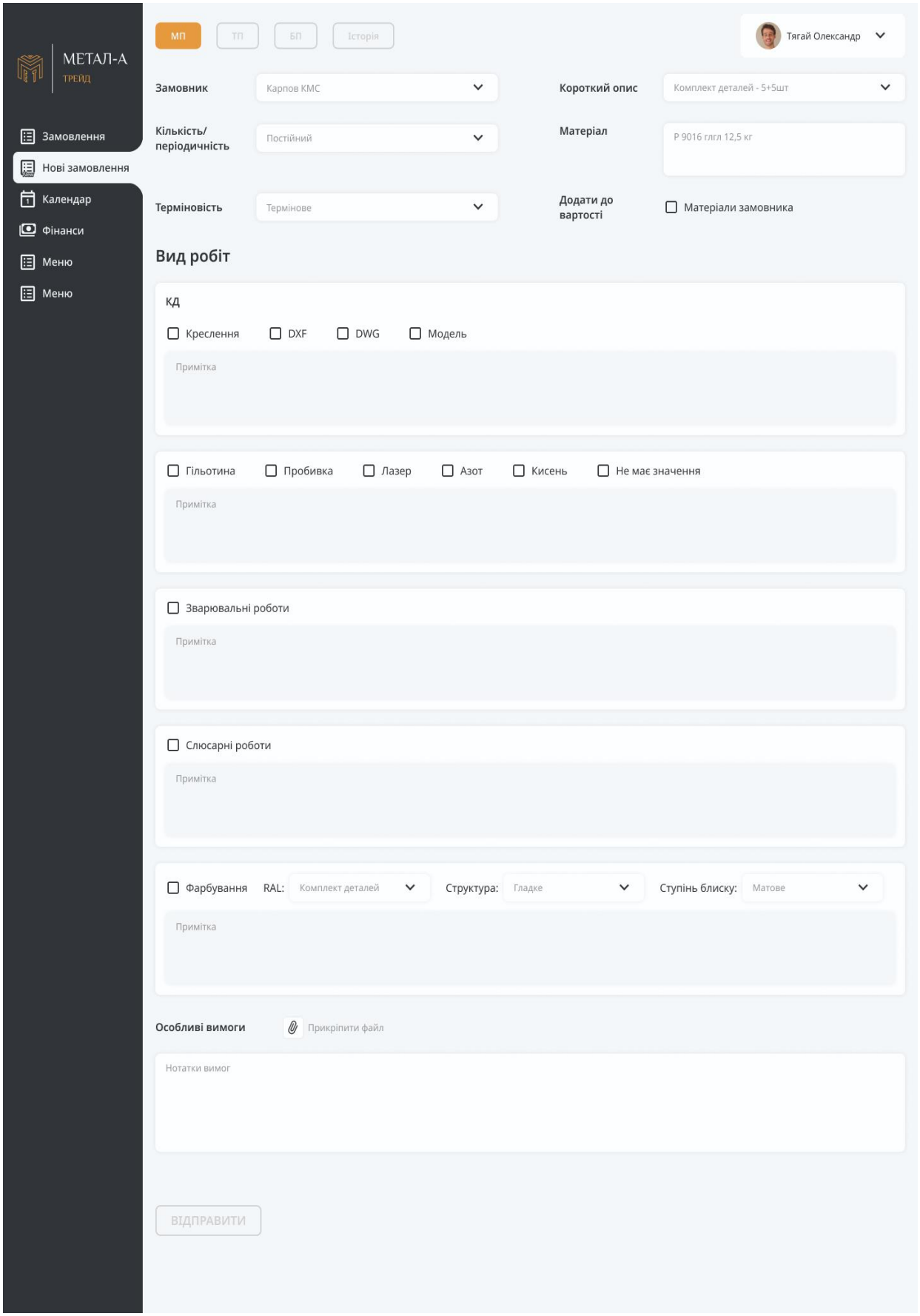


Рисунок 2.8 – Форма заповнення менеджерського прорахунку

Інтерактивність форми забезпечується через динамічне відображення опцій у випадаючих списках, як показано на Рис. 2.9, що спрощує процес вибору параметрів замовлення.

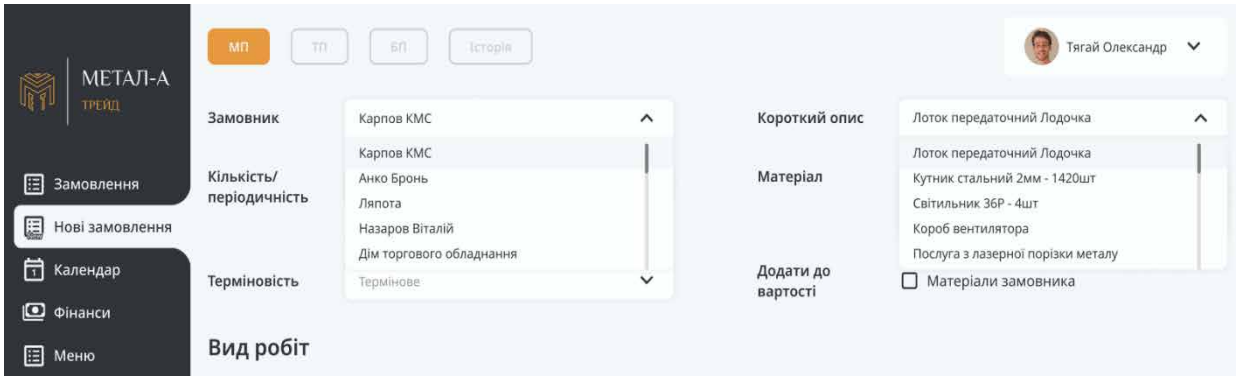


Рисунок 2.9 – Випадаючий список з варіантами вибору

Система також забезпечує можливість редагування існуючих замовлень через відповідний інтерфейс, представлений на Рис. 2.10, що дозволяє вносити зміни у параметри замовлення на різних етапах обробки.

Форма технічного прорахунку (Рис. 2.11) використовується технологами для введення технічних параметрів замовлення, включаючи розрахунок необхідних матеріалів, вибір обладнання та визначення технологічної схеми виробництва. Форма забезпечує структуроване введення всіх необхідних технічних даних для коректної реалізації замовлення.

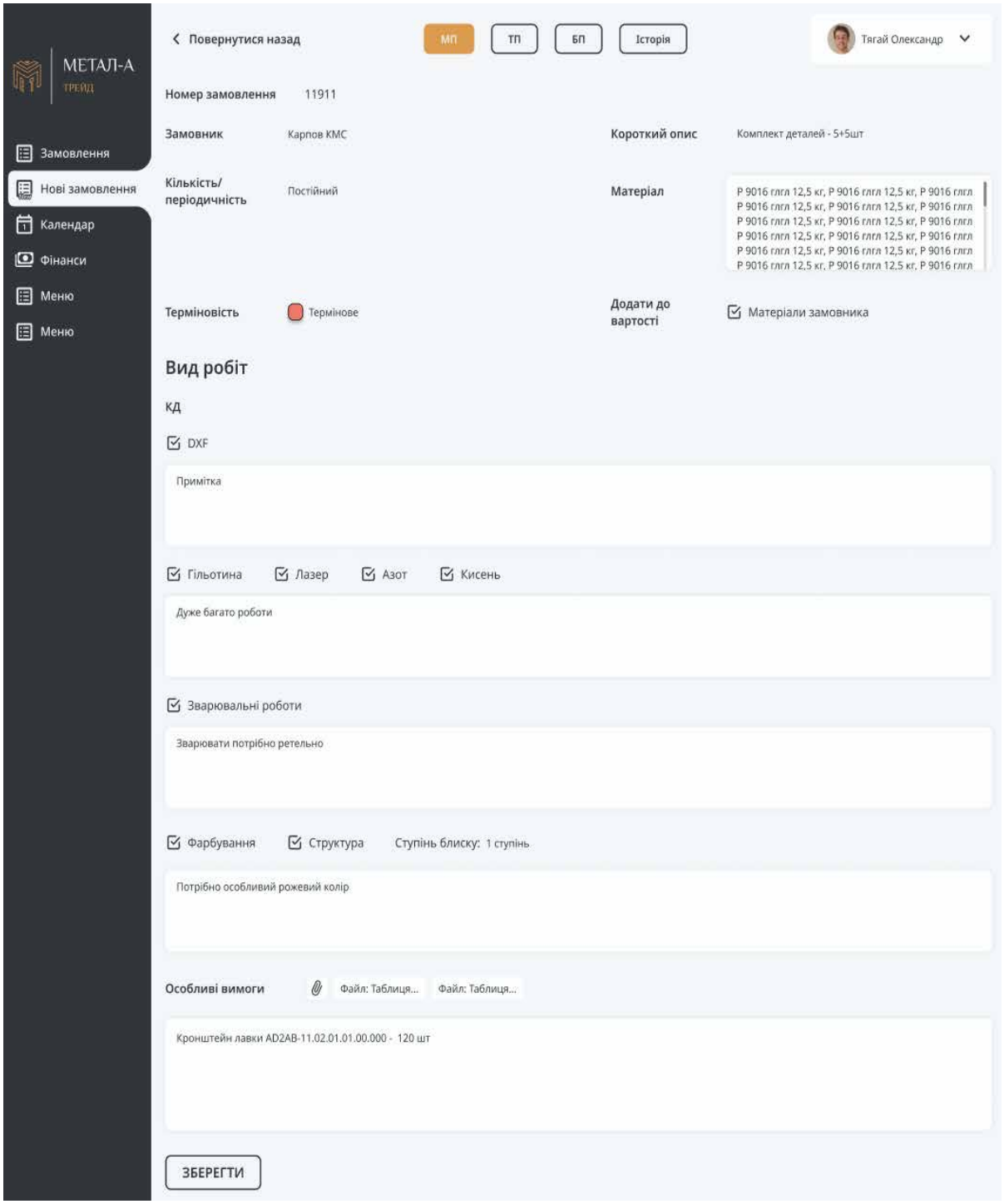


Рисунок 2.10 – Інтерфейс редагування існуючого замовлення

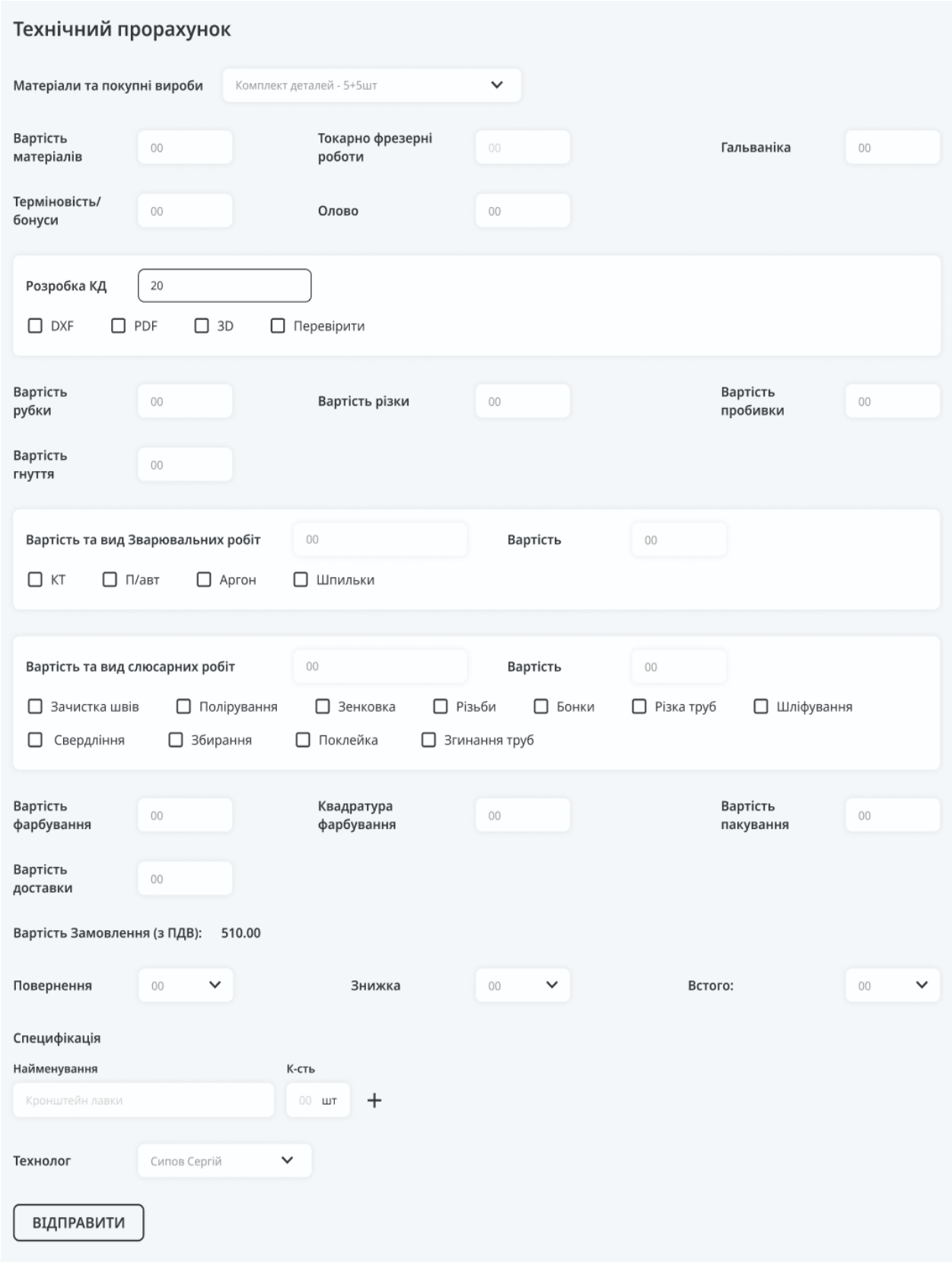


Рисунок 2.11 – Форма заповнення технічного прорахунку

Форма бухгалтерського прорахунку (Рис. 2.12) використовується бухгалтерами для формування кінцевої вартості замовлення на основі даних,

введених менеджером та технологом. Форма включає поля для розрахунку вартості матеріалів, праці, амортизації обладнання, додаткових витрат та комерційної маржі, з автоматичним обчисленням загальної суми.

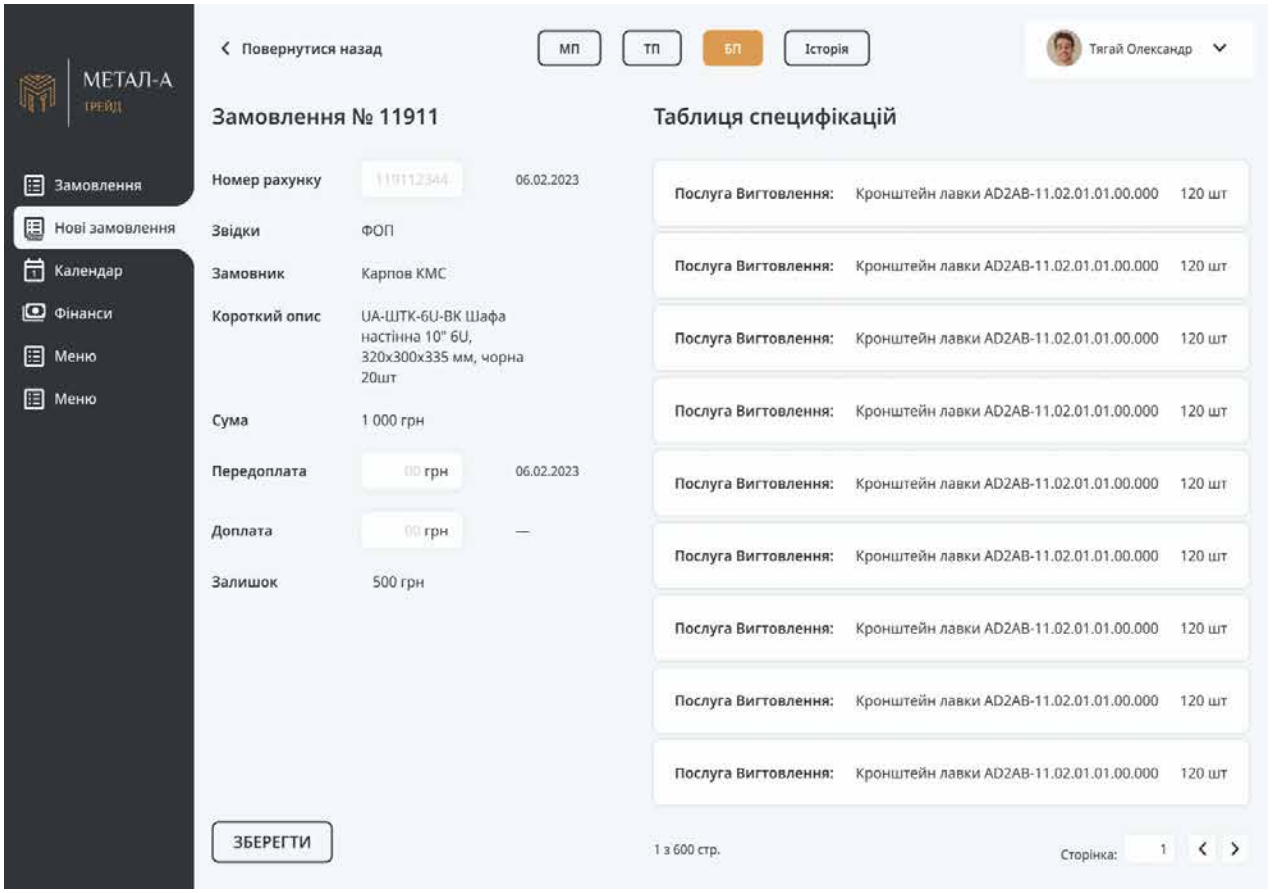


Рисунок 2.12 – Форма заповнення бухгалтерського прорахунку

Всі форми реалізують механізми валідації введених даних, забезпечують інтерактивний зворотний зв'язок та оптимізують процес введення інформації, що підвищує ефективність роботи користувачів системи та мінімізує ризик виникнення помилок.

2.5 Обґрунтування розробки системи та її переваги

Розробка веб-системи управління замовленнями для підприємств металообробної галузі обумовлена об'єктивною необхідністю подолання системних проблем, характерних для традиційних підходів до управління

виробничими процесами. Аналіз актуальних викликів, з якими стикаються підприємства даного сектору, демонструє превалювання таких проблемних аспектів, як ручне введення даних, втрата або дублювання інформації, складність координації між функціональними підрозділами та недостатній рівень автоматизації бізнес-процесів. Компаративний аналіз існуючих програмних рішень, зокрема ProShop ERP, MRPeasy та Microsoft Dynamics 365, виявив наявність певних функціональних та економічних обмежень, що актуалізують розробку спеціалізованої системи, яка інтегрує переваги існуючих продуктів та нівелює їх недоліки.

2.5.1 Обґрунтування доцільності розробки системи

Доцільність розробки власної веб-системи управління замовленнями обумовлена низкою факторів. Вирішення галузево-специфічних проблем є одним з таких факторів. Поточний підхід на основі Excel-таблиць генерує високий ризик помилок, характеризується відсутністю централізованого контролю та повільною передачею даних між функціональними підрозділами. Розроблена система автоматизує ці процеси, забезпечуючи єдине інформаційне середовище для всіх користувачів різних функціональних ролей.

Інтеграція оптимальних практик досягається через аналіз існуючих рішень (ProShop ERP, MRPeasy, Microsoft Dynamics 365), який демонструє, що кожна система має певні конкурентні переваги: глибока інтеграція з виробничими процесами у ProShop ERP, хмарна легкість MRPeasy, масштабованість Microsoft Dynamics 365. Розроблена система інтегрує ці переваги, адаптуючи їх до специфічних потреб металообробних підприємств.

Сучасний технологічний фундамент забезпечується через використання Angular для побудови інтуїтивного та реактивного фронтенду, Spring Boot для надійного та масштабованого бекенду, MongoDB для гнучкого зберігання

даних, що дозволяє створити рішення з високим рівнем надійності, масштабованості та інтеграційного потенціалу.

2.5.2 Конкурентні переваги розробленої системи

Розроблена веб-система управління замовленнями характеризується низкою конкурентних переваг порівняно з існуючими рішеннями. Гнучкість та адаптивність досягається тим, що на відміну від ProShop ERP, що фокусується на локальній інтеграції з виробничими процесами, розроблена система реалізує хмарну архітектуру з мультирольовим доступом, що забезпечує адаптивність до різних масштабів бізнесу, від малих підприємств до великих виробничих комплексів. Це суттєво розширює цільову аудиторію порівняно з MRPeasy, яка має функціональні обмеження для великих виробництв.

Оптимізована автоматизація досягається тим, що хоча Microsoft Dynamics 365 пропонує потужний аналітичний функціонал (Power BI, Copilot AI), вона характеризується складністю налаштування та високою вартістю впровадження. Розроблена система забезпечує частковий автоматизований прорахунок матеріалів та ресурсів (як MRPeasy), але з більш чіткою сегментацією функціональних ролей та мінімізацією впливу людського фактору.

Простота та доступність проявляється у тому, що на відміну від Microsoft Dynamics 365, що вимагає значних ресурсів для впровадження та підтримки, розроблена система характеризується інтуїтивним інтерфейсом (завдяки Angular) та не потребує тривалого навчання персоналу, як ProShop ERP для технологів.

Безпека та масштабованість забезпечуються використанням OAuth 2.0 для автентифікації та шифрування даних, що забезпечує вищий рівень безпеки порівняно з базовими версіями Excel або MRPeasy. MongoDB забезпечує

ефективне масштабування бази даних при зростанні обсягів замовлень, що перевершує можливості локальних рішень ProShop ERP.

Централізований контроль є перевагою розробленої системи, оскільки на відміну від розрізнених Excel-файлів або послідовної передачі даних у аналогічних системах, розроблена система пропонує єдину точку доступу до статусів замовлень, історії змін та аналітичної інформації, що суттєво підвищує транспарентність та ефективність управління.

Таким чином, розроблена веб-система управління замовленнями інтегрує оптимальні аспекти існуючих рішень, нівелюючи їх функціональні обмеження. Система орієнтована на оптимізацію внутрішніх бізнес-процесів, мінімізацію часових витрат на рутинні операції та підвищення продуктивності підприємства, пропонуючи сучасне, безпечне та масштабоване рішення, адаптоване до специфіки металообробної галузі.

Висновки до розділу 2

Проведене дослідження архітектурно-технологічних аспектів проєктування веб-системи управління замовленнями для підприємств металообробної галузі дозволяє сформулювати низку важливих висновків, що визначають концептуальні засади та практичну імплементацію розробленого рішення.

Аналіз предметної області продемонстрував актуальність та об'єктивну необхідність розробки спеціалізованої інформаційної системи для оптимізації процесів управління виробничими замовленнями на металообробних підприємствах. Ідентифіковано ключові проблеми традиційних підходів, включаючи ручне введення даних, фрагментарність інформаційних потоків, недостатню координацію між функціональними підрозділами та відсутність централізованого моніторингу статусів замовлень.

Архітектурна концепція системи базується на мультирольовому підході з чітким розмежуванням функціональних повноважень між адміністраторами,

менеджерами, технологами та бухгалтерами. Розроблена діаграма варіантів використання (Use Case) та таблиця статусів замовлень забезпечують візуалізацію функціональних взаємозв'язків та процесуальної логіки обробки замовлень у системі.

Технологічний фундамент веб-системи сформовано на основі сучасного стеку технологій, що включає Angular для реалізації клієнтської частини, Java Spring Boot для розробки серверної частини та MongoDB для управління даними. Дана комбінація забезпечує оптимальний баланс між надійністю, продуктивністю, масштабованістю та зручністю розробки.

Комунікаційна модель між фронтендом та бекендом реалізована через REST API, що забезпечує статичність, ресурсну орієнтованість, стандартизацію HTTP-методів, кешування та багаторівневість інформаційної взаємодії. Безпека комунікації забезпечується через HTTPS, OAuth 2.0, рольовий доступ та валідацію даних. Інтерфейс користувача розроблено з дотриманням принципів ергономічності, інтуїтивності та функціональної ефективності. Реалізовано спеціалізовані форми для кожного етапу обробки замовлення, включаючи менеджерський, технічний та бухгалтерський прорахунки, а також табличні інтерфейси для моніторингу та управління замовленнями.

Компаративний аналіз розробленої системи з існуючими рішеннями (ProShop ERP, MRPeasy, Microsoft Dynamics 365) виявив конкурентні переваги запропонованого підходу, включаючи гнучкість, оптимізовану автоматизацію, простоту використання, високий рівень безпеки та централізований контроль.

Практична імплементація веб-системи управління замовленнями дозволяє суттєво оптимізувати бізнес-процеси металообробних підприємств, підвищити продуктивність праці персоналу, мінімізувати ризики помилок та забезпечити високий рівень обслуговування клієнтів. Розроблена система інтегрує оптимальні аспекти існуючих рішень, нівелюючи їх функціональні обмеження, та пропонує сучасне, безпечне та масштабоване рішення, адаптоване до специфіки металообробної галузі.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СИСТЕМИ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ

3.1 Код програми та імплементація функціональності

Програмна реалізація веб-системи управління замовленнями ґрунтується на використанні сучасного фреймворку Angular версії 16+, який з релізу 15 запровадив інноваційний підхід до конфігурації без необхідності застосування класів модулів (NgModule). Архітектурне рішення системи базується на ApplicationConfig API, що становить інтегральну частину сучасних структурних змін Angular та суттєво спрощує конфігураційні процеси застосунку. Замість традиційної імплементації AppModule (модуля додатку), архітектура системи передбачає використання файлу app.config.ts, в якому сконцентровано всю конфігураційну логіку провайдерів, маршрутизації, HTTP-запитів, авторизації та налаштувань стану через NgRx Store. Такий підхід забезпечує підвищення рівня декларативності архітектури та оптимізує процеси підтримки програмного продукту.

Розглянемо детальніше ключові елементи конфігураційного файлу app.config.ts, який відповідає за налаштування параметрів застосунку:

```
typescript
import { ApplicationConfig, inject, provideAppInitializer, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';
import { routes } from '@app/app.routes';
import { provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';
import { OAuthService, provideOAuthClient } from 'angular-oauth2-oidc';
import { environment } from '../environments/environment';
import { provideStore } from '@ngrx/store';
import { provideStoreDevtools } from '@ngrx/store-devtools';
import { initializeOAuth } from '@app/core/app-management/auth-manager/oauth-initializer.factory';
import { initializeIcons } from '@app/core/app-management/svg-icon-manager/svg-icon-manager';
import { MatIconRegistry } from '@angular/material/icon';
import { DomSanitizer } from '@angular/platform-browser';
```

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideHttpClient(withInterceptorsFromDi()),
    provideOAuthClient({ resourceServer: { allowedUrls: [environment.apiServerUrl], sendAccessToken: true } }),
    provideStore(),
    provideStoreDevtools({ maxAge: 25, logOnly: environment.production }),
    provideAppInitializer(() => initializeOAuth(inject(OAuthService))()),
    provideAppInitializer(() => initializeIcons(inject(MatIconRegistry), inject(DomSanitizer))()),
  ],
};
```

Аналіз структурних компонентів конфігураційного файлу дозволяє ідентифікувати такі ключові елементи: provideZoneChangeDetection — інструмент конфігурації оптимізації зони змін у DOM-структурі, який забезпечує підвищення продуктивності обробки подій інтерфейсу користувача; provideRouter(routes) — механізм реєстрації маршрутизатора, що забезпечує функціональність навігації між різними представленнями застосунку та реагування на зміни URL; provideHttpClient(withInterceptorsFromDi()) — компонент підключення HTTP-клієнта з використанням DI-інтерсепторів, необхідних для функціонування OAuth2-авторизації та здійснення запитів до серверної частини; provideOAuthClient(...) — елемент конфігурації клієнта авторизації за протоколом OAuth2 з імплементацією обмежень доступу лише до дозволених URL-адрес, що забезпечує підвищення рівня безпеки системи; provideStore() — компонент ініціалізації сховища стану застосунку з використанням архітектури NgRx Store, що дозволяє централізовано керувати станом системи; provideStoreDevtools(...) — інструмент інтеграції засобів розробки для зручного відлагодження стану, функціональність якого автоматично

деактивується в продакшн-режимі; provideAppInitializer(...) — механізм асинхронної ініціалізації критичних компонентів системи, включаючи авторизацію (OAuth) та бібліотеку SVG-іконок, що використовуються в інтерфейсі користувача.

Маршрутизація в Angular представляє собою потужний інструментарій, що забезпечує імплементацію зручної та логічно структурованої навігації в односторінкових веб-застосунках. Функціональність маршрутизації відповідає за відображення релевантних компонентів при зміні URL-адреси, дозволяє реалізувати багаторівневу структуру представлень, обробку параметрів маршрутів та імплементацію механізмів захисту доступу. Фреймворк Angular інтегрує вбудовану систему маршрутизації, що характеризується високим рівнем гнучкості та масштабованості.

У розробленій веб-системі управління замовленнями маршрутизація виконує фундаментальну роль, забезпечуючи навігацію між функціональними модулями: перегляд активних замовлень, створення нових замовлень, проведення технічного та менеджерського прорахунків. Архітектурно маршрутизація інтегрована зі станом застосунку через NgRx, що дозволяє оптимізувати процес завантаження даних шляхом ініціювання лише необхідних частин стану при переході до конкретного представлення.

Розглянемо ключові елементи конфігурації маршрутизації в файлі app.routes.ts:

```
import { ApplicationConfig, inject, provideAppInitializer,
provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';
import { routes } from '@app/app.routes';
import { provideHttpClient, withInterceptorsFromDi } from
'@angular/common/http';
import { OAuthService, provideOAuthClient } from 'angular-oauth2-oidc';
import { environment } from '../environments/environment';
import { provideStore } from '@ngrx/store';
import { provideStoreDevtools } from '@ngrx/store-devtools';
import { initializeOAuth } from '@app/core/app-management/auth-manager/oauth-
```

```
initializer.factory';
import { initializeIcons } from '@app/core/app-management/svg-icon-
manager/svg-icon-manager';
import { MatIconRegistry } from '@angular/material/icon';
import { DomSanitizer } from '@angular/platform-browser';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
provideRouter(routes),
    provideHttpClient(withInterceptorsFromDi()),
```

Основні частини даного файлу включають:

— **provideZoneChangeDetection** — налаштовує оптимізацію зони змін у DOM (можна вимкнути, якщо використовується zone-less підхід).

— **provideRouter(routes)** — реєстрація маршрутизатора, що забезпечує навігацію між сторінками застосунку.

— **provideHttpClient(withInterceptorsFromDi())** — підключення HTTP клієнта з DI-інтерсепторами, необхідними для OAuth2.

— **provideOAuthClient(...)** — конфігурація клієнта авторизації по протоколу OAuth2 з обмеженням доступу лише до дозволених URL.

— **provideStore()** — ініціалізація сховища стану застосунку за допомогою NgRx Store.

— **provideStoreDevtools(...)** — підключення інструментів розробника для зручного дебагу стану (у продакшн-режимі вимикається).

— **provideAppInitializer(...)** — асинхронна ініціалізація: авторизації (OAuth), SVG-іконок, які використовуються в інтерфейсі.

Аналіз імплементації маршрутизації дозволяє виокремити такі архітектурні аспекти: головна сторінка (path: '') реалізує автоматичне

перенаправлення на представлення списку замовлень, що знаходяться у виробництві; маршрут «orders» відповідає за відображення активних замовлень, що знаходяться на різних етапах виробничого циклу; для кожного маршруту імплементовано механізм лінивого завантаження компонентів (loadComponent), що забезпечує оптимізацію початкового обсягу застосунку та підвищує швидкість завантаження; у контексті кожного маршруту надаються релевантні провайдери стану та ефектів (NgRx), що дозволяє гнучко управляти залежностями та забезпечувати завантаження лише необхідних сегментів стану.

Функціональність loadComponent, імплементована в Angular 14, забезпечує можливість лінивого завантаження окремих компонентів безпосередньо, без необхідності створення відокремленого модуля. Даний підхід є інтегральною частиною нової модульної архітектури Angular — Standalone Components, що дозволяє розробляти застосунки без необхідності використання традиційних NgModule. Таким чином, Angular здійснює завантаження компонента виключно при переході користувача за відповідним маршрутом, що оптимізує використання ресурсів системи.

Файлова структура застосунку організована відповідно до принципів компонентної архітектури та функціонального розподілу, що забезпечує високий рівень модульності та підтримуваності коду. Основна структура проєкту включає наступні ключові директорії: коренева папка src містить основний програмний код проєкту, а папка app інтегрує ключові модулі та компоненти системи; директорія core відповідає за імплементацію базової логіки, пов'язаної з низькорівневими аспектами функціонування застосунку, включаючи автентифікацію, авторизацію та системні сервіси; папка header інкапсулює компонент заголовка застосунку, що забезпечує навігаційний інтерфейс та відображення системної інформації; директорія orders із структурованими підкаталогами (in-production-orders, models, new-orders, services, store, validators) реалізує функціональність управління замовленнями, включаючи компоненти, моделі даних, сервіси та логіку валідації; папка shared

акумулює загальні ресурси для повторного використання в різних частинах системи, включаючи компоненти, директиви, моделі, сервіси та сховища даних; директорія side-panel відповідає за імплементацію компонента бокової панелі, що забезпечує додаткові функціональні можливості інтерфейсу; папка environments інтегрує конфігураційні налаштування для різних середовищ розробки та експлуатації системи, що дозволяє гнучко адаптувати параметри залежно від контексту використання.

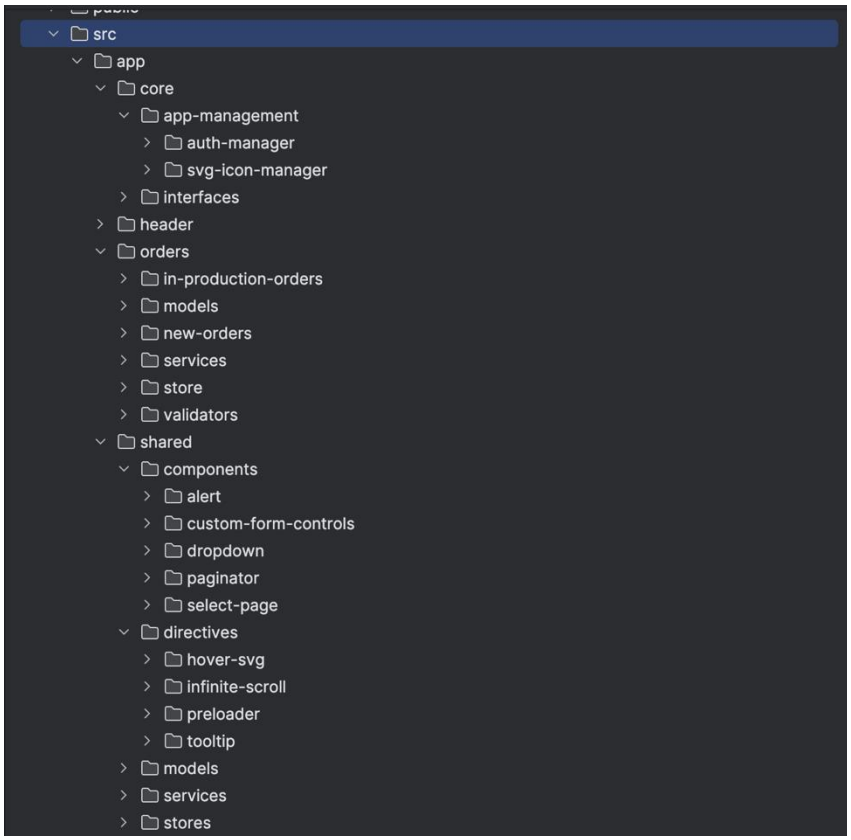


Рисунок 3.1 – Файлова структура веб-системи управління замовленнями

Розглянемо ключові аспекти імплементації компонента OrderComponent, що відповідає за відображення та управління списком замовлень:

typescript

```
export class OrderComponent implements OnInit {

  public ngOnInit(): void {
    this.dispatchGetOrderPage();
  }

  public changePageSize(size: number): void {
```

```
    this.pageSize = size;
    this.dispatchGetOrderPage();
  }

  public getOrderPage(page: number): void {
    this.orderPage = page; // need for correct sort work
    this.dispatchGetOrderPage();
  }

  public sortOrder(): void {
    this.sortType = this.getNextSortType();
    this.dispatchGetOrderPage();
  }

  public searchByColumn(): void {
    this.dispatchGetOrderPage();
  }

  public togglePanel(orderId: string, isOpen: boolean): void {
    !isOpen ? this.store.dispatch(openPanel({ orderId })) :
this.store.dispatch(closePanel({ orderId }));
  }
```

OrderComponent представляє собою Angular-компонент, що реалізує функціональність відображення списку замовлень у системі управління. Компонент інтегрований зі сховищем NgRx Store для забезпечення ефективного управління станом даних. Основні функціональні аспекти компонента включають: завантаження та відображення списку замовлень з можливістю оновлення даних; імплементацію механізмів сортування та фільтрації для оптимізації процесу пошуку інформації; реалізацію пагінації для забезпечення навігації між сторінками результатів; функціональність відображення деталізованої інформації про замовлення та завантаження

специфікацій; управління станом відображення панелі з детальною інформацією про замовлення.

Розглянемо детальніше метод getOrders, що забезпечує взаємодію з серверною частиною для отримання даних про замовлення:

typescript

```
public getOrders(
    page: number = DEFAULT_PAGE_NUMBER,
    size: number = DEFAULT_PAGE_SIZE,
    sortType: SortType = SortType.DEFAULT,
    searchValue: string = '',
    tab: string = APP_ROUTES.ORDERS
): Observable<Page<OrderResponse>> {
    const statuses: string =
        tab === APP_ROUTES.ORDERS
            ?
`status=in=(${OrderStatus.PAID},${OrderStatus.NON_PAID},${OrderStatus.SET_OFF})`
            :
`status=in=(${OrderStatus.WAITING},${OrderStatus.COUNTING},${OrderStatus.SENDING})`;

    let params: HttpParams = new HttpParams()
        .set('page', page.toString())
        .set('size', size.toString())
        .set('sort', `execution_date,${sortType}`)
        .set('query', statuses);

    if (searchValue !== '') {
        params = params.set('query', `${statuses};customer=='*${searchValue}*'`);
    }

    return
this.http.get<Page<OrderResponse>>(`${environment.apiServerUrl}/test/test`, {
params });
}
```

Метод getOrders є структурним компонентом сервісу, що забезпечує взаємодію з бекендом через HTTP-запити. Функціональне призначення методу полягає у формуванні та відправленні запиту на отримання списку замовлень з урахуванням параметрів пагінації, сортування, фільтрації та категорії замовлень. Метод приймає набір параметрів, включаючи: page — номер сторінки для пагінації (за замовчуванням: значення константи DEFAULT_PAGE_NUMBER); size — кількість записів на сторінці (за замовчуванням: значення константи DEFAULT_PAGE_SIZE); sortType — тип сортування (наприклад, за зростанням або спаданням); searchValue — текстовий параметр для фільтрації замовлень за назвою клієнта; tab — параметр, що визначає категорію замовлень для відображення (активні або нові).

Метод формує HTTP-запит з відповідними параметрами та повертає Observable об'єкт, що містить відповідь серверної частини у форматі Page<OrderResponse>. Цей метод виступає ключовим елементом забезпечення даними компонентів, що відображають списки замовлень (OrderComponent, NewOrderComponent), та реалізує функціональність пагінації, сортування та фільтрації даних.

Specifications забезпечує отримання списку специфікацій (деталізованої інформації про компоненти виробу) для конкретного замовлення. Функціональність методу використовується при відображенні вкладених таблиць та реалізації механізму безкінечної прокрутки з динамічним завантаженням даних. Метод приймає параметри orderNumber (ідентифікатор замовлення) та page (номер сторінки для пагінації) та повертає Observable об'єкт, що містить сторінку з специфікаціями.

Метод createManagerCalculation забезпечує створення нового замовлення на основі менеджерського прорахунку. Функціональність методу реалізує відправку HTTP-запиту типу POST з об'єктом managerCalculation, що містить дані форми, заповненої менеджером. Метод повертає Observable об'єкт, що містить відповідь серверної частини у форматі OrderResponse.

3.2 Аналіз компонента менеджерського прорахунку

Компонент менеджерського прорахунку (managerial-calculate.component.ts) реалізує функціональність формування та редагування замовлень менеджером системи. Ключові аспекти імплементації компонента включають функціональне призначення: компонент забезпечує функціональність введення базової інформації про замовлення (клієнт, матеріал), вибору типу робіт, завантаження файлів, управління полями для фарбування, валідації введених даних та відправки інформації на сервер; ініціалізацію форми managerialForm: структура форми включає вкладені групи полів: customer, short_description, urgency, periodicity — базові поля з імплементованими механізмами валідації, material — поля для вводу назви матеріалу та встановлення індикатора «матеріал клієнта», work_type — типи робіт (конструкторські, загальні, зварювальні, малярні), special_requirements — поле для додаткових вимог та масив attachments (файли); життєвий цикл ngOnInit: при ініціалізації компонента активується обробник для блоку фарбування, що забезпечує динамічну активацію/деактивацію відповідних полів залежно від стану прапорця present. Якщо компоненту передано ідентифікатор замовлення, ініціюється запит на отримання даних з серверної частини для відображення інформації про існуюче замовлення; управління блоком «Фарбування»: метод listenToPaintingBlock() забезпечує моніторинг змін стану поля painting.present та викликає функцію setEditableStateForPaintingBlock(checked), що активує або деактивує пов'язані поля (note, texture, sheen, RAL) залежно від встановленого стану; функціональність завантаження та видалення файлів: компонент імплементує методи uploadFiles(event: Event) для додавання нових файлів у FormArray з уникненням дублікатів та очищенням поля вводу після вибору, та removeFile(index: number) для видалення файлу за індексом з масиву.

3.3 Архітектура програми

Архітектура веб-системи управління замовленнями реалізована з використанням фреймворку Angular, що забезпечує модульність, масштабованість та зручність підтримки програмного продукту. Концептуальна архітектура застосунку базується на принципі feature-based підходу, що передбачає структурну організацію функціональних компонентів системи у вигляді окремих модулів з внутрішньою ієрархічною структурою. Додатково, архітектура включає спільні (shared) та системні (core) модулі для оптимізації повторного використання логіки, сервісів та компонентів.

Основні структурні елементи архітектури включають модуль ядра (core/), що інтегрує загальні сервіси та менеджери, які використовуються глобально в застосунку: auth-manager/ — імплементує логіку автентифікації та авторизації користувачів, svg-icon-manager/ — забезпечує функціональність динамічного завантаження та управління SVG-іконками, interfaces/ — містить глобальні інтерфейси, що використовуються у різних модулях, header/ — реалізує компонент заголовка з навігацією, доступний у всіх представленнях; функціональний модуль управління замовленнями (orders/) є ключовим модулем, що реалізує логіку створення, редагування та відображення замовлень: new-orders/ — підмодуль для створення нових замовлень, що включає: form-view/ — шаблон форми створення замовлення, managerial-calculate/ — компонент для менеджерського прорахунку, technical-calculate/ — компонент технічного прорахунку (матеріали, технології), new-order-table/ — компонент таблиці для відображення нових замовлень, in-production-orders/ — підмодуль для відображення замовлень, що перебувають на етапі виробництва, models/ — типи даних, специфічні для функціональності замовлень; глобальні сервіси (services/) включають компонент order.service.ts, що відповідає за HTTP-взаємодію з серверною частиною для операцій створення, читання, оновлення та видалення (CRUD) замовлень; модуль управління станом (store/) реалізує уніфікований механізм управління станом

застосунку з використанням бібліотеки NgRx. Структура модуля включає: order.actions.ts — визначення набору можливих дій зі станом, order.reducers.ts — імплементація логіки обробки змін стану, order.effects.ts — реалізація побічних ефектів, включаючи асинхронні запити, order.selectors.ts — селектори для доступу до сегментів стану, order.state.ts — специфікація структури стану, спеціалізовані підмодулі для managerial-calculate та technical-calculate; спільні елементи (shared/) формують модуль для повторного використання компонентів у різних частинах системи: components/ — спільні UI-компоненти, directives/ — директиви для розширення функціональності елементів, services/ — утилітарні сервіси (наприклад, форматування даних), models/ — загальні моделі даних, використовувані в різних модулях; компонент бічної панелі (side-panel/) реалізує логіку та візуальне представлення бокової панелі інтерфейсу, що розширює функціональні можливості системи; користувацькі валідатори (validators/) інтегрують кастомні механізми валідації для Angular Reactive Forms, що забезпечують верифікацію введених користувачем даних відповідно до специфіки предметної області.

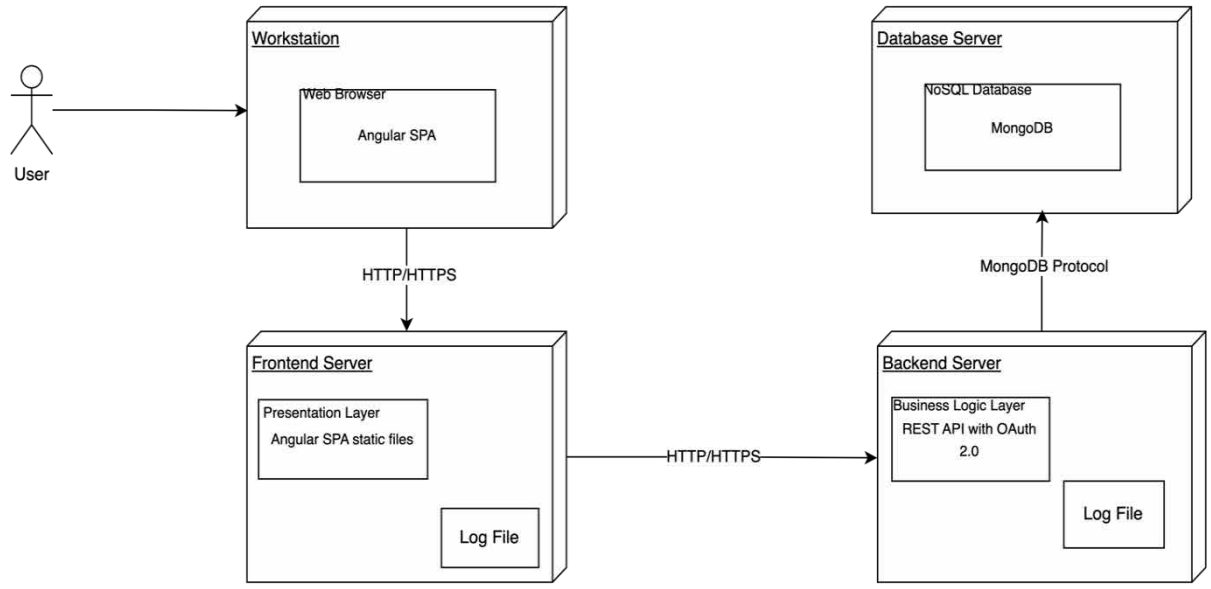


Рисунок 3.2 – Архітектурна структура веб-системи

Архітектурний підхід, імплементований у системі, забезпечує низку значних переваг: масштабованість: кожна функціональна одиниця характеризується високим рівнем ізоляції, що дозволяє ефективно розширювати або модифікувати її функціональність без впливу на інші модулі системи; оптимізація повторного використання коду: впровадження shared і core модулів суттєво знижує рівень дублювання логіки та забезпечує консистентність програмного коду; спрощення процесів супроводу: чітка структурна організація сприяє ефективному орієнтуванню в кодовій базі та оптимізує процеси внесення змін та розширення функціональності; централізоване управління станом: використання архітектури NgRx забезпечує передбачуваність та ефективний контроль над станом застосунку, що оптимізує процеси діагностики та усунення потенційних проблем.

3.4 Архітектура системи

Архітектурна концепція веб-системи управління замовленнями базується на клієнт-серверній парадигмі з чітким розмежуванням відповідальності між компонентами. Діаграма системної архітектури візуалізує процеси взаємодії користувача з системою та функціональні взаємозв'язки між різними архітектурними шарами.

Основні компоненти архітектури включають: користувач (User): первинна точка ініціації взаємодії з системою, що здійснює доступ до застосунку через веб-браузер; робоча станція (Workstation): клієнтський пристрій з встановленим веб-браузером (Google Chrome, Firefox тощо), що забезпечує функціональність відображення та взаємодії з Angular Single Page Application (SPA). Angular обрано як ключову технологію для реалізації клієнтської частини через його потужні можливості для створення сучасних, реактивних інтерфейсів користувача; фронтенд-сервер (Frontend Server): серверний компонент, відповідальний за доставку інтерфейсу користувача. Включає Presentation Layer (шар презентації), що інтегрує Angular SPA та

статичні файли (HTML, CSS, JavaScript). Додатково, реалізовано механізм логування (Log File) для запису подій та помилок, що виникають на фронтенді; бекенд-сервер (Backend Server): серверний компонент, що реалізує обробку даних та бізнес-логіку застосунку. Включає Business Logic Layer (шар бізнес-логіки), що функціонує через REST API з імплементацією OAuth 2.0 для автентифікації. Також включає окремий механізм логування для фіксації подій серверної частини; сервер бази даних (Database Server): серверний компонент для зберігання та управління даними системи. Архітектурно інтегрує NoSQL базу даних MongoDB, обрану завдяки її оптимальній адаптивності для роботи з гнучкими структурами даних, характерними для предметної області. Бекенд-сервер взаємодіє з MongoDB через відповідний протокол для операцій зберігання та отримання даних.

Функціональна взаємодія компонентів архітектури: користувач ініціює доступ до застосунку через браузер, який формує HTTP/HTTPS-запити до фронтенд-сервера для отримання інтерфейсу; фронтенд-сервер доставляє клієнтську частину (Angular SPA) та забезпечує подальшу комунікацію з бекенд-сервером через HTTP/HTTPS для отримання або відправки даних; бекенд-сервер обробляє запити, взаємодіє з базою даних MongoDB для операцій з даними та формує відповіді, що передаються фронтенд-компоненту; уся архітектурна взаємодія захищена механізмами OAuth 2.0, а системи логування на фронтенді та бекенді забезпечують моніторинг та діагностику функціонування системи.

3.5 Інтерфейс користувача

Інтерфейс користувача веб-системи управління замовленнями розроблено з фокусом на забезпечення оптимальної інтуїтивності, функціональності та ергономічності. Ключові компоненти інтерфейсу включають сторінку замовлень, замовлення з відкритими специфікаціями,

детальну інформацію про замовлення та форму заповнення менеджерського прорахунку.

Сторінка замовлень представляє основний інтерфейсний компонент, що відображає перелік замовлень у табличному форматі з інформацією про статус, клієнта, термін виконання та інші ключові параметри. Інтерфейс забезпечує функціональність сортування, фільтрації та пагінації для ефективного управління великими обсягами даних.

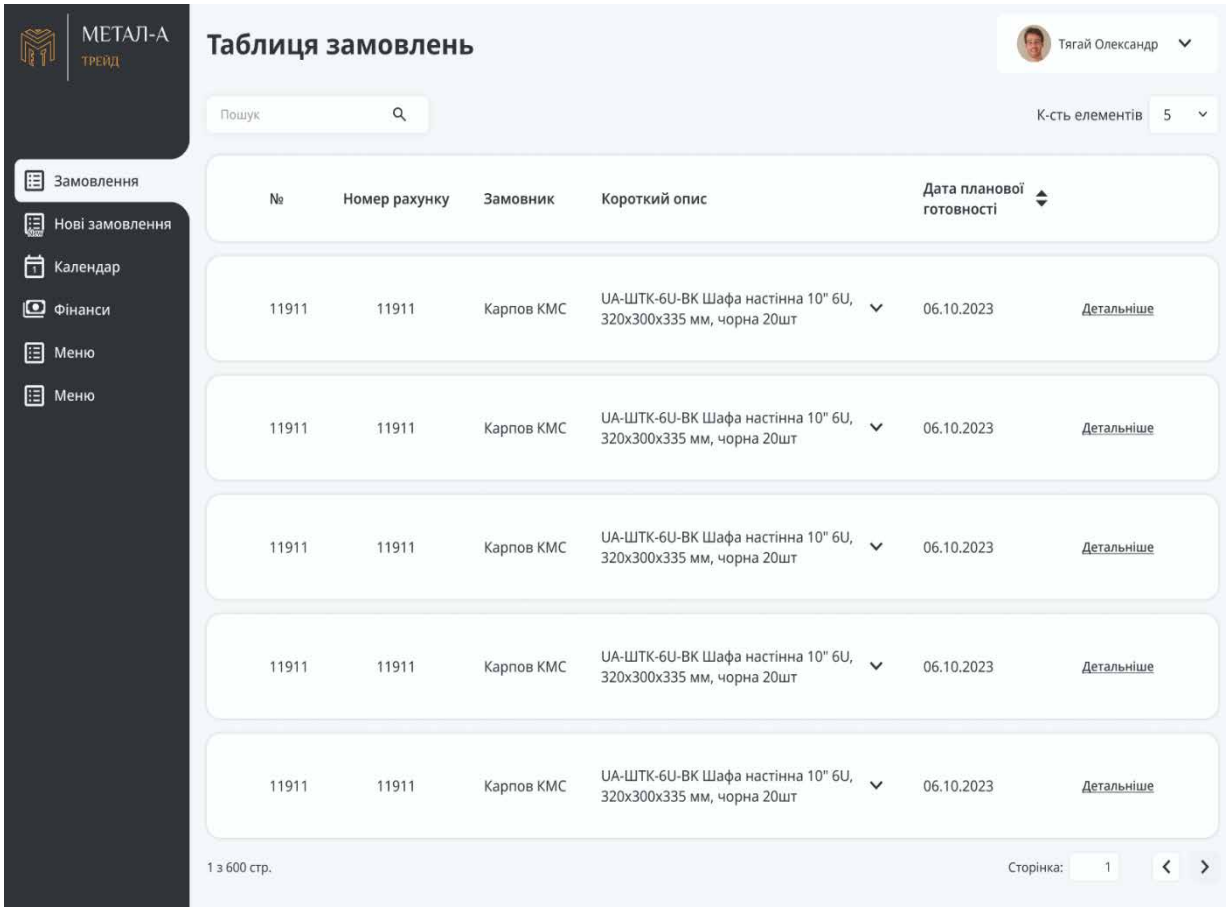
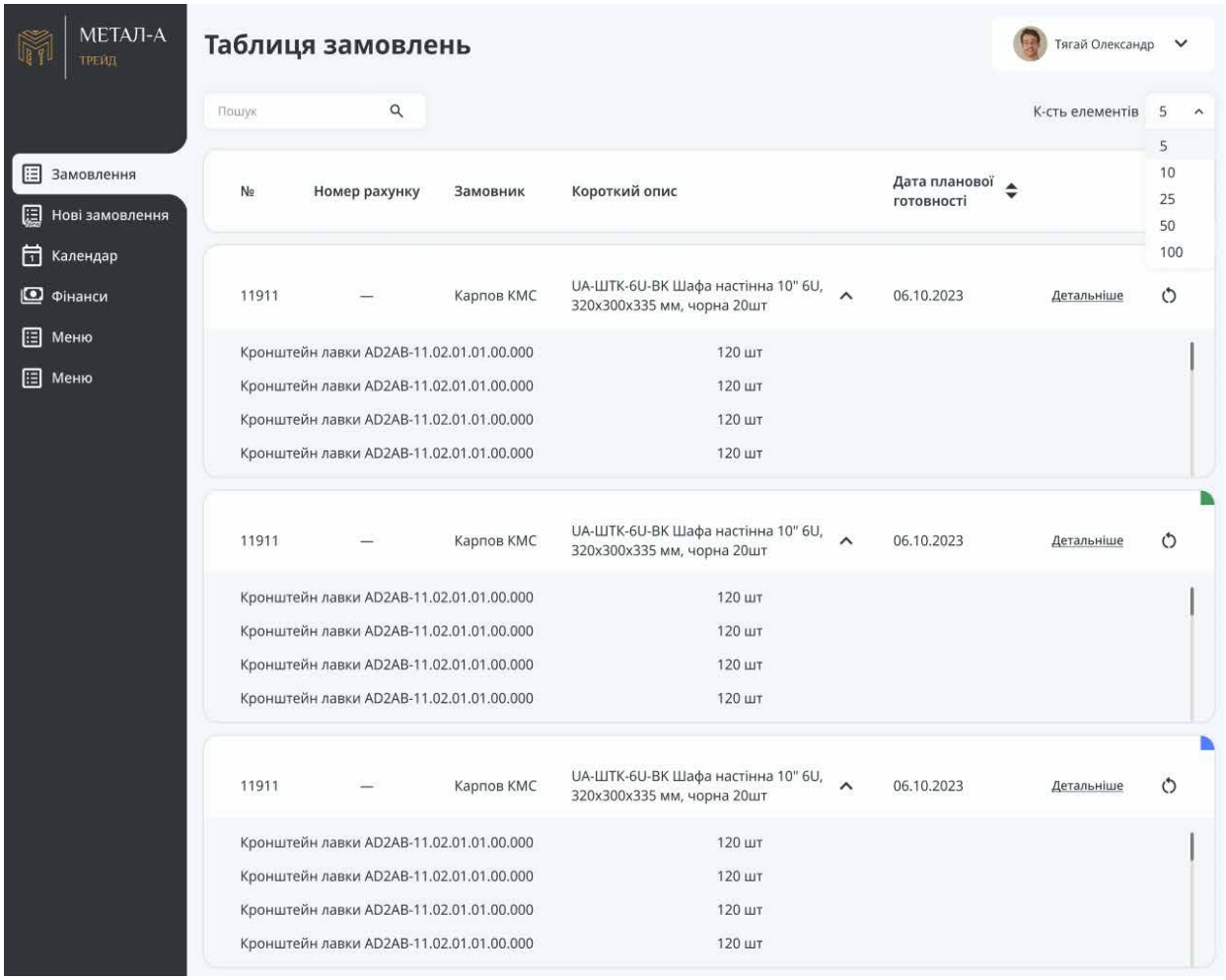


Рисунок 3.4 – Сторінка замовлень веб-системи



Таблиця замовлень

№	Номер рахунку	Замовник	Короткий опис	Дата планової готовності
11911	—	Карпов КМС	UA-ШТК-6U-ВК Шафа настінна 10" 6U, 320x300x335 мм, чорна 20шт	06.10.2023
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
11911	—	Карпов КМС	UA-ШТК-6U-ВК Шафа настінна 10" 6U, 320x300x335 мм, чорна 20шт	06.10.2023
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
11911	—	Карпов КМС	UA-ШТК-6U-ВК Шафа настінна 10" 6U, 320x300x335 мм, чорна 20шт	06.10.2023
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	
Кронштейн лавки AD2AB-11.02.01.01.00.000			120 шт	

Рисунок 3.5 – Замовлення з відкритими специфікаціями

Інтерфейс замовлень з відкритими специфікаціями представляє розширену версію представлення замовлень, що включає відображення деталізованої інформації про компоненти замовлення (специфікації). Інтерфейс реалізує функціональність динамічного розгортання/згортання деталізованої інформації для оптимізації візуального сприйняття даних.

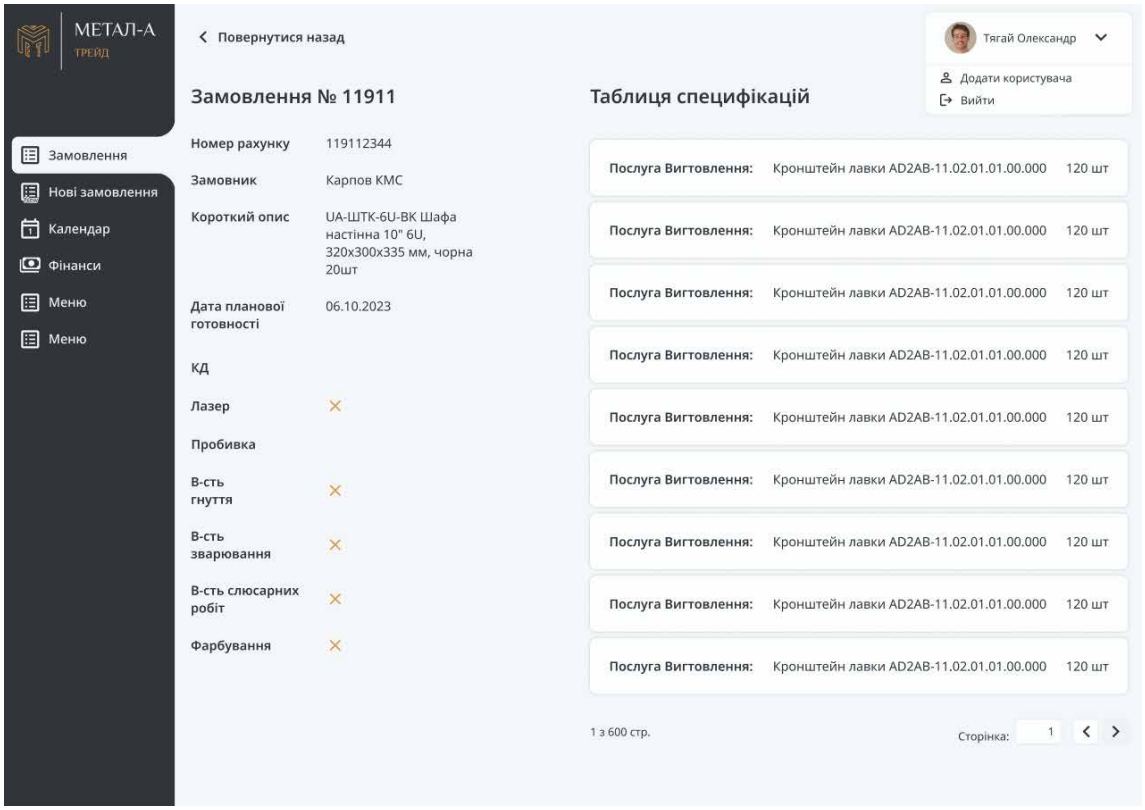


Рисунок 3.6 – Детальна інформація про замовлення

Інтерфейс детальної інформації про замовлення забезпечує комплексне відображення всієї доступної інформації про конкретне замовлення, включаючи дані про клієнта, параметри замовлення, статус виконання та історію змін.

Форма заповнення менеджерського прорахунку представляє структурований інтерфейс для введення базової інформації про нове замовлення, включаючи дані про клієнта, терміновість, періодичність, тип матеріалу та вимоги до обробки. Форма реалізує динамічну валідацію введених даних та інтерактивну зміну доступності полів залежно від обраних опцій.

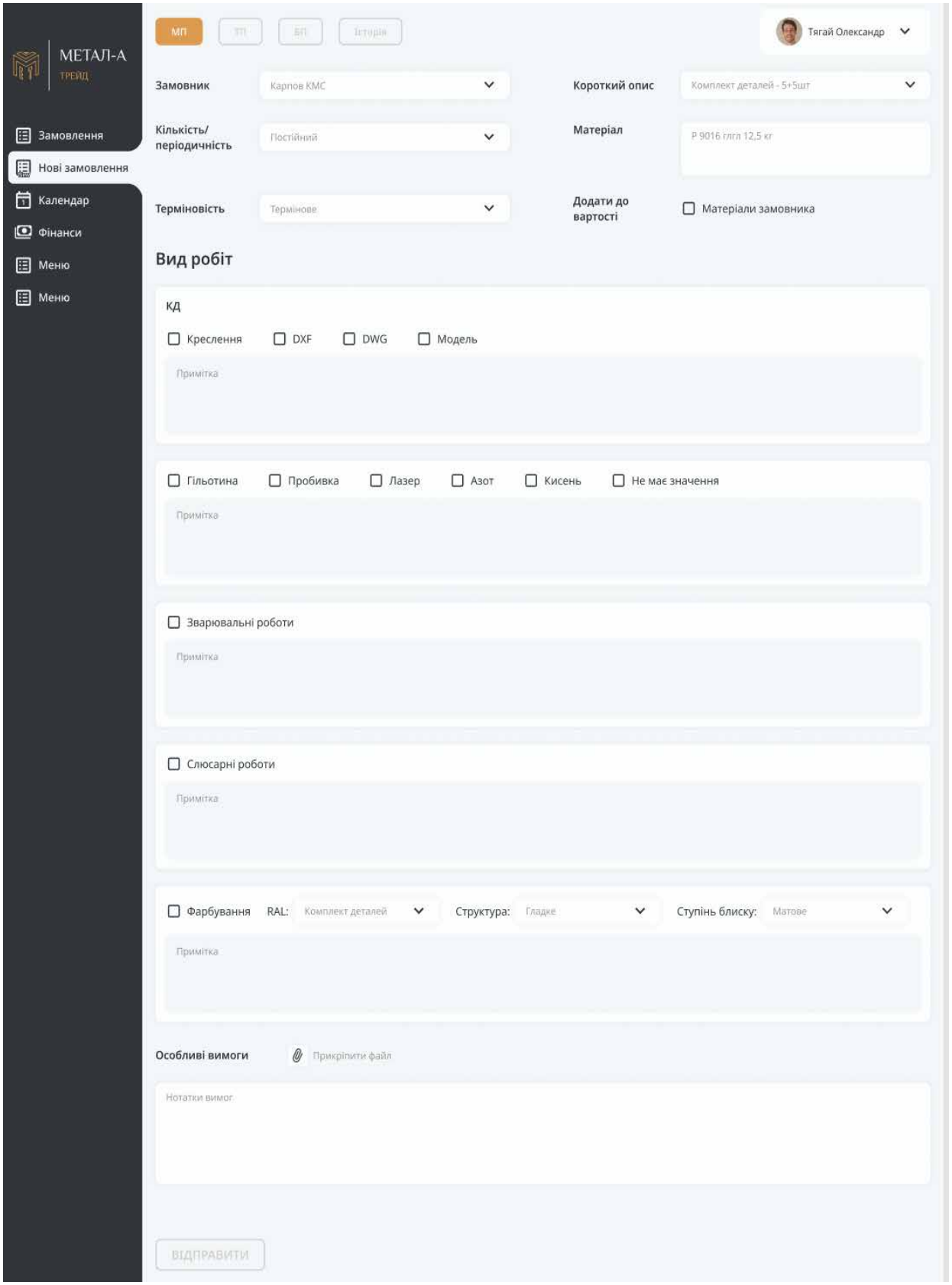


Рисунок 3.7 – Форма заповнення менеджерського прорахунку

3.6 Тестування роботи програми

Процес тестування веб-системи управління замовленнями включав верифікацію функціональності ключових компонентів та перевірку відповідності системи заданим вимогам. Основні аспекти тестування включали верифікацію функціональності автентифікації, тестування функціональності відображення нових замовлень та верифікацію вкладки активних замовлень.

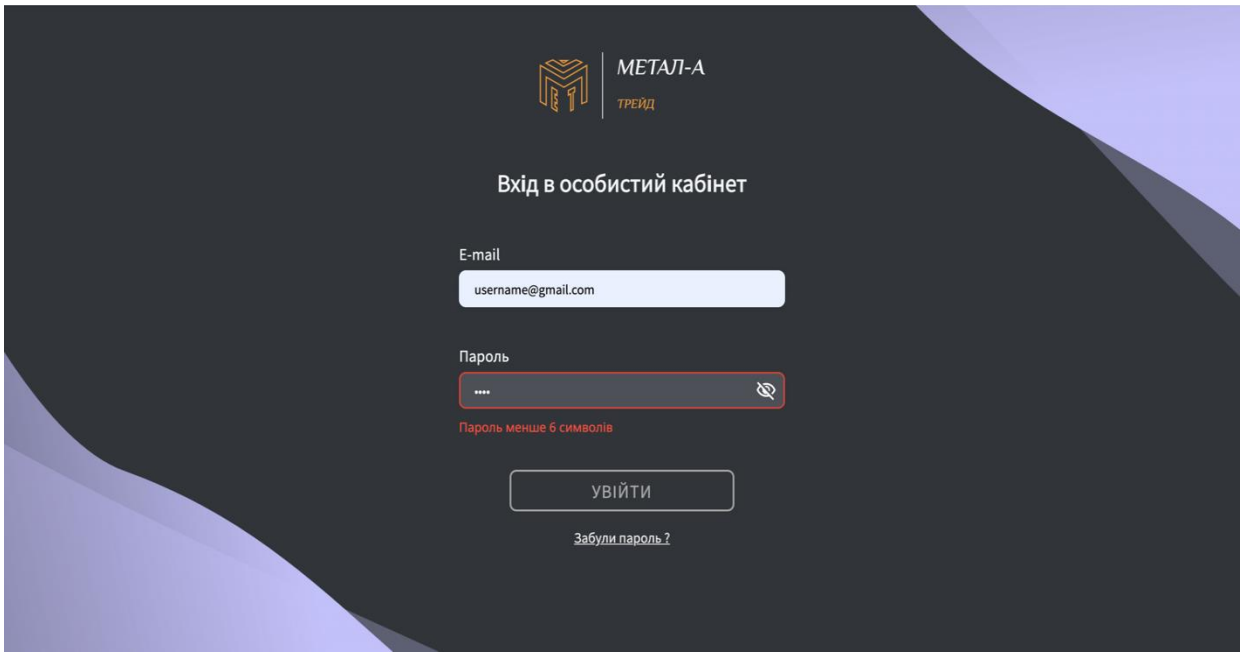


Рисунок 3.8 – Тестування функціональності автентифікації та обробки помилок

Тестування процесу авторизації користувачів та механізмів обробки помилок при введенні некоректних облікових даних підтвердило коректність роботи системи автентифікації та інформативність повідомлень про помилки.

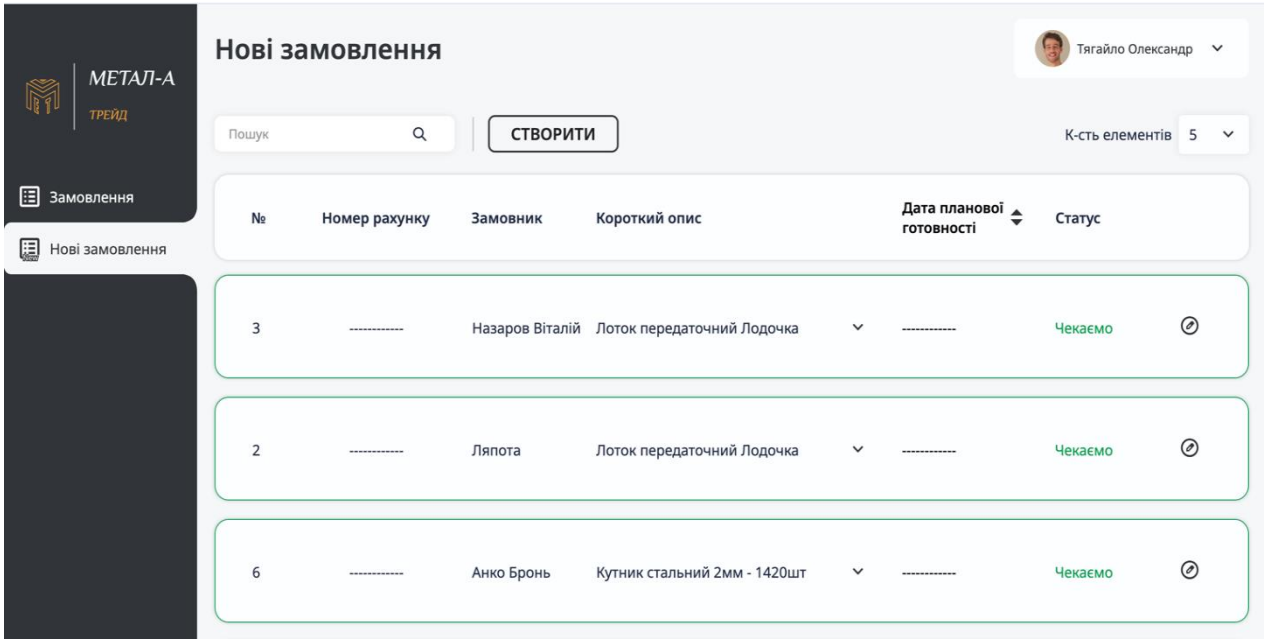


Рисунок 3.9 – Тестування функціональності відображення нових замовлень

Перевірка коректності роботи вкладки «Нові замовлення», що відображає замовлення, створені менеджером зі статусом «Чекаємо», продемонструвала коректність відображення даних та функціональність фільтрації та сортування.

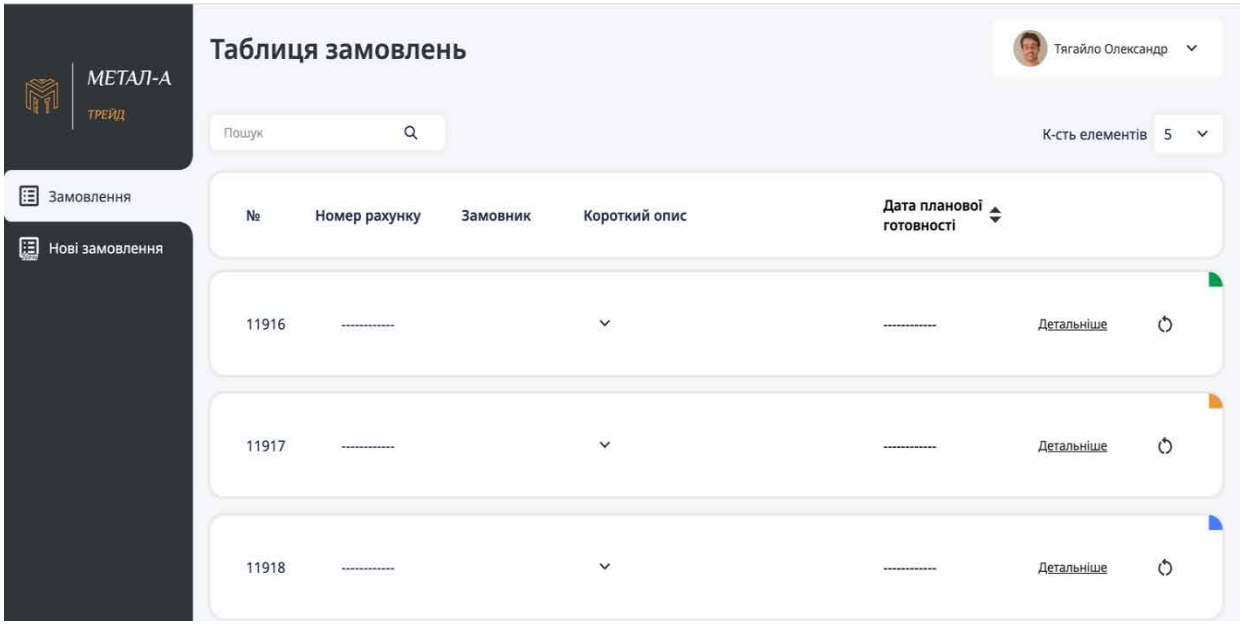


Рисунок 3.10 – Тестування вкладки активних замовлень

Тестування функціональності відображення замовлень, що знаходяться в роботі, підтвердило коректність фільтрації за статусами та забезпечення оновлення даних при зміні стану системи.

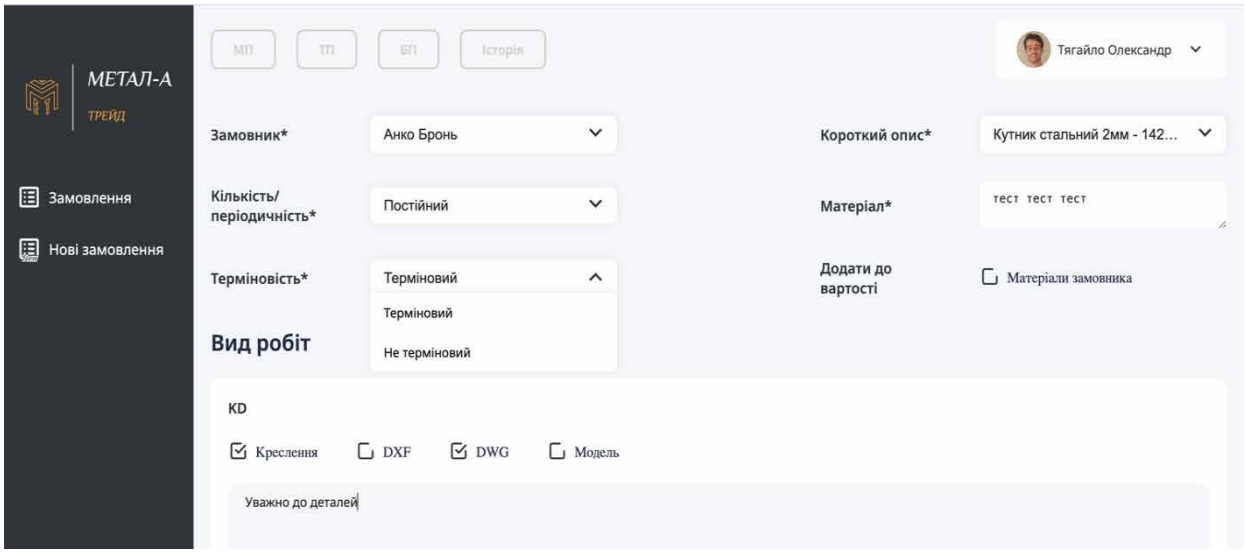


Рис. 3.11 – Тестування форми заповнення менеджерського прорахунку

Верифікація функціональності форми менеджерського прорахунку продемонструвала коректність роботи механізмів валідації, динамічної активації полів та збереження введених даних.

Висновки до розділу 3

У третьому розділі магістерської роботи здійснено безпосередню програмну реалізацію веб-системи управління замовленнями для підприємства з виробництва металу. Основна увага була сконцентрована на практичній імплементації функціональності, визначеної у попередніх розділах, а також на верифікації коректності функціонування системи відповідно до заданих функціональних та нефункціональних вимог.

В рамках розробки системи реалізовано комплекс ключових технологічних рішень, що забезпечують надійність, масштабованість та ергономічність користувацького досвіду. Імплементовано ефективну архітектуру програми, що базується на принципах багаторівневої організації з

чітким розмежуванням презентаційного рівня (Angular), логіки бізнес-процесів (Java Spring Boot) та механізмів зберігання даних (MongoDB). Використання клієнт-серверної архітектури забезпечило незалежність між клієнтською та серверною компонентами, що суттєво підвищує гнучкість процесів розгортання та подальшого супроводу системи.

Детально описано логічну структуру взаємодії компонентів системи з використанням схематичних діаграм, фрагментів програмного коду та конфігураційних файлів, що демонструють сучасний підхід до побудови інтерфейсів користувача, реалізації маршрутизації, інтеграції сховища стану (NgRx) та забезпечення взаємодії з програмним інтерфейсом (API).

Впроваджено прогресивний стек фронтенд-технологій, де клієнтська частина системи реалізована з використанням Angular 16+, що забезпечило можливість використання інноваційних підходів, таких як Standalone Components, ApplicationConfig API та відмова від NgModules, що суттєво спрощує структурну організацію проєкту. Компонентна архітектура Angular забезпечує ефективне масштабування функціональності, RxJS імплементує реактивну обробку подій та HTTP-запитів, NgRx реалізує централізоване управління станом застосунку з можливістю відстеження та контролю життєвого циклу даних, SCSS використовується для стилізації інтерфейсу з підтримкою змінних, міксинів та вкладених структур, маршрутизація забезпечує навігаційну функціональність, а REST API реалізує стандартизований підхід до взаємодії з серверною частиною.

Імплементовано комплексну логіку управління замовленнями з програмною реалізацією функціональності створення, перегляду та обробки замовлень. Реалізовано форму для менеджерського прорахунку, що забезпечує можливість врахування складності, періодичності, терміновості та специфіки матеріалів. Структурна організація форми включає вкладені групи полів, кастомні механізми валідації та логіку динамічної активації/деактивації полів (наприклад, при включенні функціональності фарбування).

Також реалізовано технологічні механізми динамічного завантаження специфікацій з підтримкою безкінечної прокрутки (infinite scroll), обробки помилок при взаємодії з API та формування параметризованих запитів до серверної частини з можливістю сортування, фільтрації та пагінації результатів.

Проведено тестування системи з демонстрацією функціональності у різних сценаріях використання, включаючи створення нового замовлення менеджером, проведення технічного прорахунку матеріалів та технологічних процесів, реалізацію фінансового розрахунку та інтеграцію замовлення до загального реєстру. Система продемонструвала стабільне функціонування при обробці запитів, забезпечуючи коректну передачу даних між компонентами та синхронізацію стану. Інтерфейс користувача зберігає ергономічність та інтуїтивність навіть при збільшенні обсягу даних, що обробляються.

Інтерфейс користувача, розроблений з використанням Angular Material та власних SCSS-стилів, забезпечує адаптивність, інтуїтивність та візуальну чистоту представлення. Створено макети та скріншоти, що відображають ключові представлення системи: таблиці замовлень, форми створення та прорахунку, панелі деталізованої інформації.

Узагальнюючи результати третього розділу, можна констатувати успішну реалізацію повноцінного веб-застосунку відповідно до заданих функціональних вимог, імплементацію архітектури, що забезпечує легкість розширення та адаптації системи до змінних потреб, впровадження сучасних підходів до управління станом, забезпечення модульності та оптимізації користувацького досвіду, а також проведення комплексного тестування з демонстрацією прикладів функціонування, що підтверджує готовність системи до практичного впровадження.

Перспективні напрямки подальшого розвитку системи включають інтеграцію аналітичного модуля зі статистичною інформацією та візуалізацією даних, забезпечення взаємодії з зовнішніми ERP-системами, реалізацію

механізмів push-сповіщень та адаптацію для мобільних пристроїв з можливістю створення окремого PWA-клієнта.

ВИСНОВКИ

Проведене дослідження дозволило реалізувати комплексне рішення для автоматизації процесів управління виробничими замовленнями на підприємствах металообробної галузі. Розроблена веб-система забезпечує оптимізацію внутрішніх бізнес-процесів, підвищення продуктивності праці персоналу та покращення якості обслуговування клієнтів.

У першому розділі магістерської роботи проведено ґрунтовний аналіз предметної області, що дозволив ідентифікувати ключові проблеми існуючих підходів до управління замовленнями на металообробних підприємствах. Визначено, що традиційні методи, базовані на використанні електронних таблиць та розрізнених програмних рішень, призводять до значних інформаційних втрат, дублювання даних та низької ефективності взаємодії між функціональними підрозділами. Компаративний аналіз існуючих програмних рішень (ProShop ERP, MRPeasy, Microsoft Dynamics 365) виявив їх функціональні обмеження та економічні бар'єри для впровадження на підприємствах малого та середнього бізнесу. На основі проведеного аналізу сформовано комплекс функціональних та нефункціональних вимог до інструментальних засобів розробки веб-системи.

Другий розділ роботи присвячено архітектурно-технологічним аспектам проєктування системи. Запропоновано багаторівневу архітектуру з чітким розмежуванням презентаційного рівня (Angular), рівня бізнес-логіки (Java Spring Boot) та рівня даних (MongoDB). Реалізовано мультирольовий підхід з диференціацією функціональних повноважень між адміністраторами, менеджерами, технологами та бухгалтерами. Розроблено процесуальний алгоритм обробки замовлень, що забезпечує послідовну передачу відповідальності між функціональними суб'єктами та контроль статусу виконання на кожному етапі. Сформовано класифікаційну структуру замовлень за типами виконання, клієнтської приналежності, пріоритетності, стану готовності даних та типу продукції.

У третьому розділі здійснено програмну реалізацію веб-системи з використанням сучасних технологій та архітектурних патернів. Клієнтська частина системи реалізована на базі Angular 16+ з імплементацією компонентної архітектури, що забезпечує модульність, масштабованість та зручність підтримки. Використано ApplicationConfig API для конфігурації застосунку, що суттєво спрощує структурну організацію проєкту. Реалізовано маршрутизацію з лінивим завантаженням компонентів та інтеграцією зі сховищем стану NgRx. Розроблено функціональні компоненти для відображення списків замовлень, деталізованої інформації та форми для менеджерського прорахунку. Імплементовано логіку взаємодії з бекендом через REST API з підтримкою пагінації, сортування та фільтрації даних.

Проведене тестування системи підтвердило її відповідність заданим функціональним та нефункціональним вимогам. Система демонструє стабільне функціонування при обробці запитів, забезпечує коректну передачу даних між компонентами та синхронізацію стану. Інтерфейс користувача зберігає ергономічність та інтуїтивність навіть при збільшенні обсягу даних, що обробляються.

Практична значимість розробленої веб-системи полягає у можливості її безпосередньої імплементації на підприємствах металообробної галузі для автоматизації процесів управління замовленнями. Система забезпечує централізований контроль життєвого циклу замовлення, оптимізацію інформаційних потоків між функціональними підрозділами, мінімізацію ризиків втрати даних та підвищення точності прогнозування термінів виконання та вартості замовлень.

Наукова новизна роботи визначається розробленим архітектурним рішенням, що забезпечує оптимальний баланс між функціональною повнотою, масштабованістю та економічною доцільністю впровадження. Запропонований підхід до класифікації замовлень та процесуальний алгоритм їх обробки враховують специфіку металообробних підприємств та забезпечують адаптивність системи до різних виробничих контекстів.

Перспективні напрямки подальшого розвитку системи включають інтеграцію аналітичного модуля зі статистичною інформацією та візуалізацією даних, забезпечення взаємодії з зовнішніми ERP-системами, реалізацію механізмів push-сповіщень та адаптацію для мобільних пристроїв з можливістю створення окремого PWA-клієнта.

Підсумовуючи результати проведеного дослідження, можна констатувати, що розроблена веб-система управління замовленнями для підприємств з виробництва металу представляє собою ефективний інструмент автоматизації бізнес-процесів, який забезпечує підвищення продуктивності праці, оптимізацію використання ресурсів та покращення якості обслуговування клієнтів. Система поєднує сучасні технологічні рішення з глибоким розумінням специфіки предметної області, що забезпечує її практичну цінність та конкурентоспроможність порівняно з існуючими програмними продуктами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Business Process Management System, BPM. URL: https://www.tadviser.ru/a/117491 (дата звернення 09.03.2024)
2. Buriak I., Nechyporenko K., Chychun V., Polianko H., Milman L. Trends in the development of management and business technology in the formation of the modern Ukrainian economy. Futurity Economics&Law. 2022. No 2(4), pp. 29–35.
3. Впровадження ERP. URL: https://ontargit.com/ua/implementation-of-erp/
4. Гарькава В. Ф., Хитрова О. А., Пшенична М. В., Орленко О. В. (2023). Тренди розвитку менеджменту та бізнес-технологій в умовах формування сучасної української економіки. Академічні Візії, 16. DOI: https://doi.org/10.5281/zenodo.7670619
5. ERP-система для малого бізнесу, переваги та ризики. URL: https://www.netsoft.com.ua/ERP-systema-dlya-maloho-biznesu.html
6. Ліпич Л.Г., Хілуха О.А., Кушнір М.А. Еволюція розвитку інформаційних систем управління підприємством. Економічний форум. 2021. Т. 1, №4. С. 85–94.
7. Україна 2030 - країна з розвинутою цифровою економікою URL: https://strategy.uifuture.org/kraina-z-rozvinutoyu-cifrovoyu-ekonomikoyu.html
8. Business Process Management System, BPM. Available at: https://www.tadviser.ru/a/117491 (accessed 09 March 2024)
9. Buriak I., Nechyporenko K., Chychun V., Polianko H., Milman L. (2022) Trends in the development of management and business technology in the formation of the modern Ukrainian economy. Futurity Economics&Law. No 2(4), pp. 29–35. (in Ukrainian)
10. Vprovadzhennia ERP. [Implementation of ERP] Available at: https://ontargit.com/ua/implementation-of-erp/ (accessed 09 March 2024).

11. Harkava V. F., Khytrova O. A., Pshenychna M. V., Orlenko O. V. (2023) Trendy rozvytku menedzhmentu ta biznes-tekhnolohii v umovakh formuvannia suchasnoi ukrainskoi ekonomiky. [Trends in the development of management and business technologies in the conditions of the formation of the modern Ukrainian economy.] Akademichni Vizii, 16. Available at: https://doi.org/10.5281/zenodo.7670619 (accessed 09 March 2024). (in Ukrainian)
12. ERP-systema dlia maloho biznesu, perevahy ta ryzyky. [ERP system for small business, advantages and risks.] Available at: https://www.netsoft.com.ua/ERP-systema-dlya-maloho-biznesu.html (accessed 10 March 2024)
13. Lipych, L., Khilukha, O., and Kushnir, M. (2021) "Evolution of the development of enterprise management information systems". [Evolution of the development of enterprise management information systems] Economic Forum, vol. 1(4), pp. 85–94. (in Ukrainian)
14. Ukraina 2030 - kraina z rozvynutoiu tsyfrovoiu ekonomikoiu. [Ukraine 2030 is a country with a developed digital economy]. Available at: https://strategy.uifuture.org/kraina-z-rozvinutoyu-cifrovoyu-ekonomikoyu.html (accessed 10 March 2024)
15. Learn C#. OFFICIAL COLLECTION: https://learn.microsoft.com/enus/users/dotnet/collections/yz26f8y64n7k07?WT.mc_id=dotnet-35129-website
16. C# programming guide https://learn.microsoft.com/enus/dotnet/csharp/programmingguide/?WT.mc_id=dotnet-35129-website
17. Object-oriented programming (C#) https://learn.microsoft.com/enus/dotnet/csharp/fundamentals/tutorials/oop?source=recommendations
18. Inheritance in C# and .NET https://learn.microsoft.com/enus/dotnet/csharp/fundamentals/tutorials/inheritance
19. Language Integrated Query (LINQ) https://learn.microsoft.com/en-us/dotnet/csharp/programmingguide/concepts/linq 6. Troelsen Andrew, Japikse Phil. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming – APress, 2022 – 1640 pp.

20. Christian Nagel Professional C# and .NET, 2021st Edition – Wrox, 2021 – 1800 pp. Додаткова література 8. Mark J. Price C# 10 and .NET 6 – Modern Cross-Platform Development – Packt Publishing, 2021 – 826 pp.
21. Joseph Albahari C# 10 in a Nutshell: The Definitive Reference – O'Reilly Media, 2022 – 1000 pp. 10.Ian Griffiths C# 10 in a Nutshell – O'Reilly Media, 2022 – 833 pp.