МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

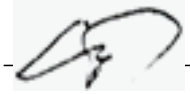
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунка для внутрішніх перевезень компанії «Agrofusion»

Виконав: студент групи 4151

 Поляк Д.О.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____  Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Поляк Денис Олександрович
(Прізвище, ім'я, по батькові)

1. Тема роботи: Вебзастосунок для внутрішніх перевезень компанії "Agrofusion"
Керівник роботи Пухалевич Андрій Володимирович
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р.
3. Вихідні дані по роботі:
4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).
5. Перелік презентаційних матеріалів 1. Титульний аркуш; 2. Завдання на кваліфікаційну роботу; 3. Вступ; 4. Аналіз практичної задачі інженерії програмного забезпечення; 5. Проєкт програмного забезпечення; 6. Результати розробки програмного забезпечення; 7. Розділ з охорони праці; 8.Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Поляк Д.О.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

АНОТАЦІЯ

У кваліфікаційній роботі розглядається задача інженерії програмного забезпечення, пов’язана з розробкою вебзастосунка для автоматизації внутрішніх перевезень компанії “Agrofusion”, яка здійснює діяльність у сфері агробізнесу. Актуальність роботи зумовлена необхідністю забезпечення ефективного управління логістичними процесами між структурними підрозділами компанії – сільськогосподарськими угіддями, переробними заводами, складами та офісами.

У процесі виконання роботи було проаналізовано вимоги до функціональності вебзастосунку, досліджено наявні програмні рішення в галузі логістики, визначено їхні обмеження у контексті агросектору. Запропоновано власну архітектуру вебзастосунка, реалізовано основні модулі вебзастосунку: реєстрації користувачів, створення та обробки замовлень, призначення виконавців, контролю статусів перевезень.

Для розробки використано стек технологій Java, Spring Boot, REST API, MySQL та сучасні засоби побудови вебінтерфейсу. Проведено тестування програмного забезпечення, оформлено повний комплект супровідної документації, включаючи технічне завдання, опис програмних модулів, інструкції для користувача та програміста, а також програму і методику випробувань.

Робота викладена на 133 сторінках, містить 65 рисунків, 11 таблиць, 5 додатків, список використаних джерел з 14 найменувань. Кваліфікаційна робота виконана українською мовою.

ABSTRACT

The thesis deals with the software engineering task related to the development of a web application for automating internal transportation of Agrofusion, a company operating in the field of agribusiness. The relevance of the work is due to the need to ensure effective management of logistics processes between the company's structural units agricultural land, processing plants, warehouses and offices.

In the course of the work, were analyzed the requirements for the system's functionality, studied existing software solutions in the field of logistics, and identified their limitations in the context of the agricultural sector. Proposed own architecture of the web application, implemented the main modules of the system: user registration, creation and processing of orders, appointment of contractors, control of transportation statuses.

The development was based on a stack of Java, Spring Boot, REST API, MySQL technologies and modern web interface development tools. The software was tested, and a full set of supporting documentation was prepared, including the terms of reference, descriptions of program modules, user and programmer instructions, as well as a test program and methodology.

The work is set out on 133 pages, contains 65 figures, 11 tables, 5 appendices, and a list of references consisting of 14 titles. The qualification work is written in Ukrainian.

ЗМІСТ

ВСТУП

У сучасному бізнес-середовищі ефективне управління внутрішніми логістичними процесами є критично важливим для забезпечення стабільної роботи підприємств. Для компаній, що працюють у сфері агробізнесу, зокрема таких як "Agrofusion" [1], організація внутрішніх перевезень є важливою складовою забезпечення безперебійного виробництва та своєчасної доставлення продукції. В кваліфікаційній роботі вирішується така задача інженерії програмного забезпечення як розробка вебзастосунка для внутрішніх перевезень компанії “Agrofusion”. Ця задача характеризується комплексністю та невизначеністю умов, так як логістична система компанії охоплює сільськогосподарські угіддя, переробні заводи, складські приміщення та адміністративні офіси, що потребує високої координації логістичних процесів між різними підрозділами компанії.

Наявне програмне забезпечення для контролю логістичних процесів орієнтоване переважно на великі логістичні компанії та не враховує специфіку аграрного сектору. Зокрема, більшість таких систем недостатньо гнучкі для адаптації до сезонних змін та особливостей виробничих процесів компанії "Agrofusion", що робить їх малопридатними для впровадження. Тому розробка власного вебзастосунка для внутрішніх перевезень компанії “Agrofusion” виглядає найбільш доцільною з урахуванням потреб підприємства.

Мета роботи: розробка вебзастосунка для внутрішніх перевезень компанії “Agrofusion”.

Для досягнення поставленої мети необхідно виконати такі завдання:

1) проаналізувати задачу інженерії програмного забезпечення з розробки вебзастосунка для внутрішніх перевезень компанії "Agrofusion";

2) проаналізувати існуюче програмне забезпечення для управління логістикою компаній;

3) поставити задачу на розробку вебзастосунка для внутрішніх перевезень компанії "Agrofusion";

4) розробити архітектуру вебзастосунка для внутрішніх перевезень компанії “Agrofusion”;

5) розробити проєкт вебзастосунка для внутрішніх перевезень компанії “Agrofusion”;

6) програмно реалізувати вебзастосунок для внутрішніх перевезень компанії “Agrofusion”;

7) виконати тестування вебзастосунка для внутрішніх перевезень компанії “Agrofusion”.

Об’єкт кваліфікаційної роботи: Процес розробки програмного забезпечення для автоматизації внутрішніх логістичних перевезень компанії "Agrofusion".

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ “AGROFUSION”

1.1 Аналіз задачі інженерії програмного забезпечення з розробки вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Комплексність та невизначеність умов практичної задачі інженерії програмного забезпечення з розробки вебзастосунка внутрішніх перевезень компанії "Agrofusion" проявляється в декількох ключових аспектах.

Багатошаровість організаційної структури. Компанія "Agrofusion" має складну організаційну структуру, що включає сільськогосподарські угіддя, переробні заводи, складські приміщення та адміністративні офіси. Кожен з цих підрозділів має власні логістичні потреби, правила та обмеження. Вебзастосунок повинен враховувати специфіку роботи кожного підрозділу, забезпечуючи при цьому єдиний підхід до управління перевезеннями.

Різноманітність типів вантажів і транспортних засобів. Компанія перевозить різні типи вантажів: від свіжих томатів, що вимагають особливих умов транспортування з контролем температури, до обладнання та запасних частин, для яких критична безпека перевезення. В основному для перевезення вантажів компанія залучає найманий транспорт що своєю чергою ускладнює контроль та облік необхідної інформації по перевезеннях.

Сезонність виробництва. Бізнес компанії "Agrofusion" має виражену сезонність. У період збору врожаю томатів обсяги перевезень значно збільшуються, що створює пікове навантаження на логістичну інфраструктуру. Вебзастосунок повинен бути здатним ефективно працювати як в періоди максимального навантаження, так і в міжсезоння, коли інтенсивність перевезень

зменшується, але з'являються нові типи логістичних операцій (наприклад, доставлення обладнання для ремонту та модернізації виробництва).

Складність бізнес-процесів. Процес управління внутрішніми перевезеннями включає багато етапів: від створення заявки на перевезення до оформлення документів про доставлення вантажу. Вебзастосунок повинен забезпечувати чітке визначення ролей, розподіл відповідальності та контроль виконання всіх етапів процесу, забезпечуючи при цьому гнучкість для врахування нестандартних ситуацій [2].

Масштабування та розширення функціоналу. З часом, коли базовий функціонал програмного забезпечення буде освоєний користувачами, може виникнути потреба в розширенні можливостей вебзастосунка. Додавання нових функцій або збільшення кількості користувачів може створити додаткове навантаження на вебзастосунок, що вимагатиме перегляду архітектурних рішень. Невизначеність щодо майбутніх потреб компанії створює додаткові виклики для розробників.

На цей момент для контролю внутрішніх перевезень у компанії "Agrofusion" використовується розрізнений набір інструментів та підходів, що не забезпечує єдиної централізованої системи управління. Основними компонентами наявного підходу можна зазначені.

Усі документи зберігаються на платформі 1С, але через відсутність централізованої платформи зростає складність контролю та знижується прозорість процесів.

Для обліку та аналізу перевезень використовуються файли Excel, що зберігаються локально на комп'ютерах відповідальних осіб або в корпоративному сховищі даних. Хоча такий підхід дозволяє структурувати дані, він має обмеження щодо одночасного доступу користувачів, контролю версій та інтеграції з іншими системами.

Система ТЛК (Тотальний Логістичний Контроль). Ця система використовується переважно для моніторингу найманого транспорту під час сезону збирання томатів.

Електронна пошта. Комунікація між різними підрозділами щодо планування та координації перевезень часто здійснюється через електронну пошту, що ускладнює відстеження історії комунікацій та прийнятих рішень.

Основні недоліки наявного підходу до управління внутрішніми перевезеннями є відсутність централізованого контролю. Інформація про перевезення зберігається в різних місцях та форматах, що ускладнює її аналіз та прийняття рішень на основі даних.

Висока ймовірність помилок. Ручне заповнення документів та багаторазове введення одних і тих самих даних у різні системи збільшує ризик виникнення помилок.

Складність відстеження статусу перевезень. Немає можливості оперативно отримувати інформацію про поточний стан вантажу, місцеперебування транспортного засобу та очікуваний час доставляння.

Обмежені аналітичні можливості. Наявні інструменти не дозволяють ефективно аналізувати дані про перевезення для оптимізації маршрутів, завантаження транспорту та інших аспектів логістичних операцій.

Обмежені можливості масштабування. Наявний підхід важко масштабувати при збільшенні обсягів перевезень або розширенні географії діяльності компанії.

1.2 Аналіз наявного програмного забезпечення для управління логістикою компаній

Аналіз ринку програмного забезпечення для управління логістикою показує, що існують комерційні рішення, які потенційно могли б задовольнити потреби компанії "Agrofusion".

MercuryGate – це система управління перевезеннями (TMS), яка допомагає компаніям оптимізувати та автоматизувати процеси транспортування вантажів [3].

На рисунку 1.1 представлено головну сторінку вебзастосунка MercuryGate.

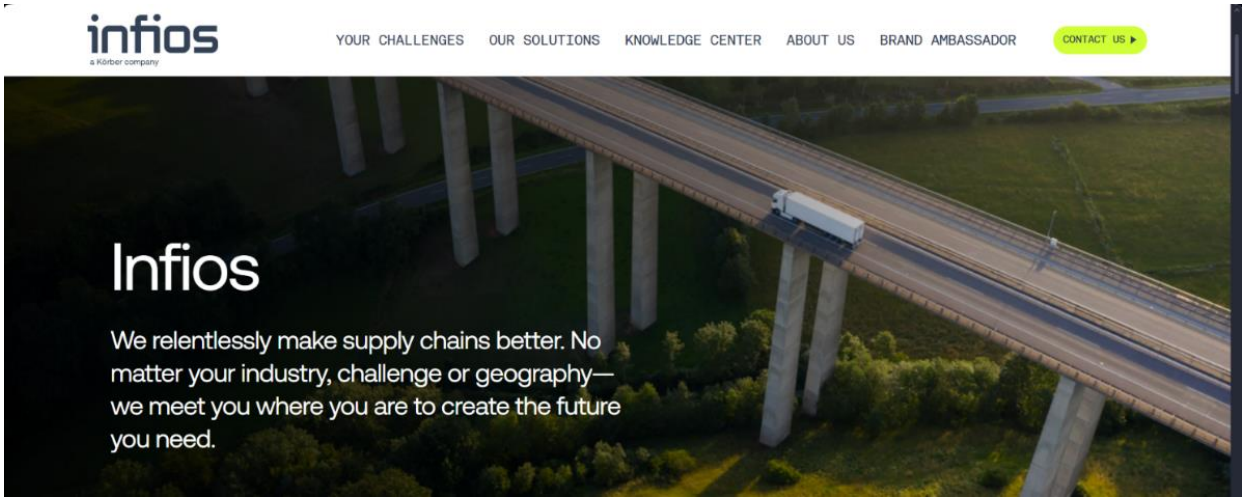


Рисунок 1.1 − Головна сторінка MercuryGate

Розробником є MercuryGate International, Inc.

Система забезпечує автоматизацію всього життєвого циклу перевезень, включаючи маршрутизацію, тарифікацію та оптимізацію логістичних процесів. Це дозволяє зменшити людський фактор та підвищити ефективність операцій.

Вбудована бізнес-аналітика надає інструменти для збору, аналізу та візуалізації даних, що сприяє ухваленню стратегічних рішень на основі перевіреної інформації.

Система легко інтегрується з іншими програмними продуктами та може масштабуватись під складні потреби бізнесу. Це дозволяє підприємствам адаптуватися до змін без суттєвих витрат на заміну ПЗ.

Також передбачено управління та планування переміщення запасів, що сприяє оптимізації витрат і забезпеченню безперервності ланцюгів постачання.

Серед переваг варто відзначити динамічність та багатофункціональність, що дозволяє налаштовувати систему під різноманітні бізнес-процеси. Також високий рівень видимості задач управління перевезеннями забезпечує прозорість і контроль на всіх етапах логістики.

Недоліком є висока вартість впровадження та технічної підтримки, що може стати бар'єром для невеликих підприємств. Крім того, система може виявитися надлишковою за функціональністю для компаній з простими потребами, що призводить до зайвих витрат.

Oracle Transportation Management (OTM) – це хмарна платформа, яка допомагає компаніям управляти всіма аспектами транспортування в межах глобального ланцюга постачань. Вона пропонує функції оперативного планування, управління флотом, моделювання логістичної мережі, цифрового асистента та машинного навчання для досягнення оптимальної роботи логістики [4].

На рисунку 1.2 представлено головну сторінку вебзастосунка Oracle Transportation Management.

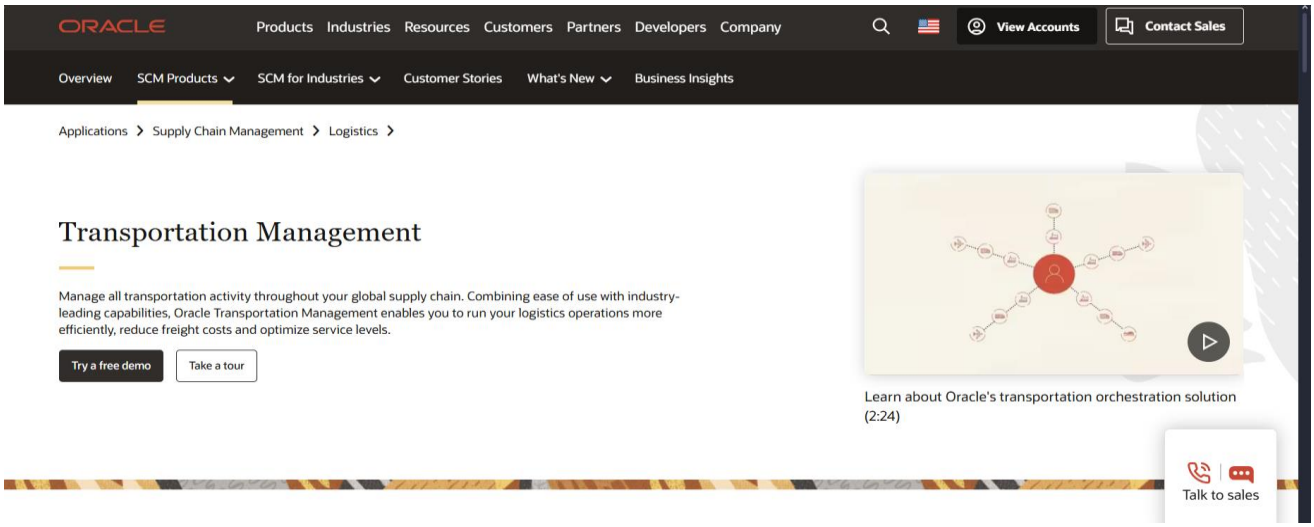


Рисунок 1.2 − Головна сторінка Oracle Transportation Management

Розробником системи є компанія Oracle Corporation. Вона пропонує рішення для оперативного планування з вбудованим механізмом оптимізації логістичних рішень, що дозволяє підвищити ефективність процесів транспортування в реальному часі.

Система має інформаційну панель, яка забезпечує зручне відстеження деталей перевезень і ключових логістичних показників, що сприяє прийняттю обґрунтованих управлінських рішень.

Передбачено функціонал для управління автопарком, що дозволяє контролювати технічний стан, використання ресурсів і ефективність роботи транспорту.

Також система підтримує управління вимогами глобального ланцюга постачання, забезпечуючи координацію і синхронізацію процесів між різними учасниками на міжнародному рівні.

Серед переваг – ефективне управління фінансовими операціями, зменшення витрат і підвищення прозорості обліку. Система добре підходить для великих компаній з розгалуженою мережею постачання у глобальному масштабі.

Недоліками є висока вартість рішення, що може бути недосяжною для невеликих компаній, а також складність впровадження та налаштування, що потребує часу та залучення досвідчених спеціалістів.

Blue Yonder – це комплексне рішення, розроблене для оптимізації транспортування та логістичних операцій у різних галузях. Це система, яка забезпечує повну видимість, планування та виконання процесів, що допомагає компаніям підвищити ефективність, знижувати витрати та покращувати рівень обслуговування в рамках своїх ланцюгів постачання [5].

На рисунку 1.3 представлено головну сторінку вебзастосунка Blue Yonder.

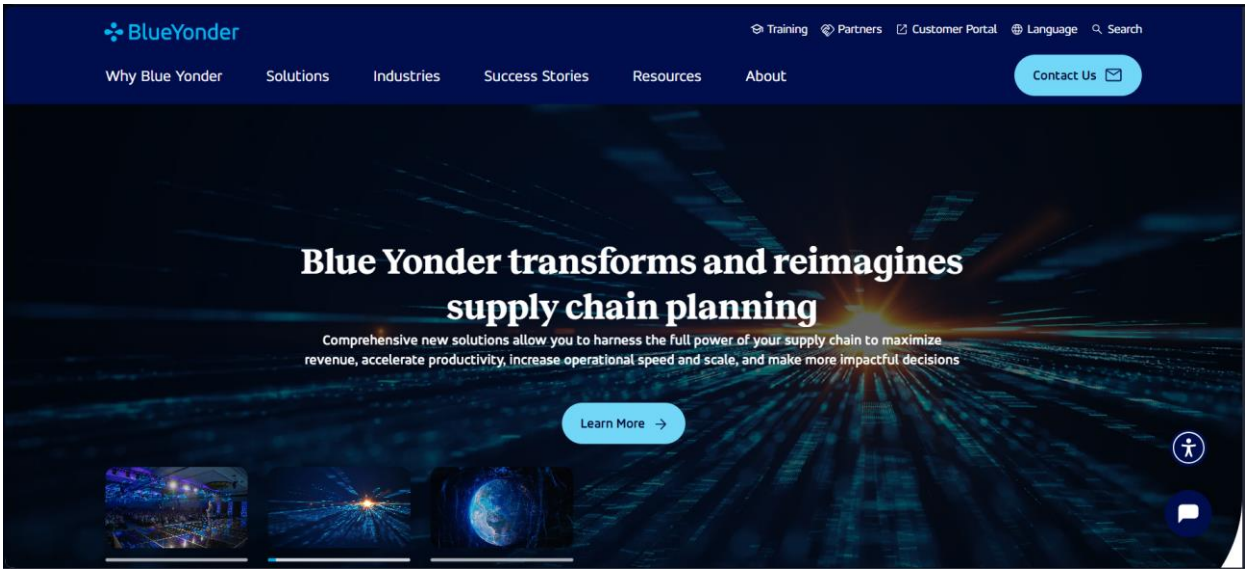


Рисунок 1.3 − Головна сторінка Blue Yonder

Blue Yonder забезпечує автоматичне прогнозування попиту з використанням алгоритмів штучного інтелекту, що дозволяє компаніям точно планувати обсяги постачань і зменшувати ризики нестач або надлишків товару.

Система підтримує глибоку інтеграцію між постачальниками, складами та транспортними компаніями, забезпечуючи ефективну координацію дій для задоволення потреб клієнтів.

Функціонал включає оптимізацію рівнів запасів з метою зменшення витрат на зберігання, підвищення точності виконання замовлень і зниження кількості помилок під час доставлення товарів.

Система підтримує роботу з різними видами транспорту автомобільним, залізничним, морським та авіаційним, а також оптимізує маршрути перевезень і вибір перевізників для забезпечення найвищої ефективності логістики.

Blue Yonder оснащена потужними аналітичними інструментами, які дають змогу збирати, аналізувати та візуалізувати дані в реальному часі, що суттєво покращує процеси прийняття рішень на всіх рівнях управління.

Інтеграція всіх етапів ланцюга постачання підвищує загальну ефективність компанії, дозволяє знизити витрати на транспортування та зберігання товарів. Система універсальна – підходить як для великих, так і для середніх підприємств у різних галузях, зокрема роздрібній торгівлі, виробництві та логістиці.

Великим плюсом є можливість адаптації системи під індивідуальні потреби, а також використання штучного інтелекту для автоматизації ключових управлінських процесів.

Серед недоліків – висока вартість впровадження та підтримки, що обмежує доступність для малих і частини середніх підприємств. Система вимагає значних інвестицій у навчання персоналу та розвиток інфраструктури.

Процес впровадження часто є складним і потребує багато часу, особливо якщо компанія має численні взаємопов’язані системи. Крім того, налаштування й оптимізація потребують залучення спеціалістів з відповідними знаннями.

Оскільки рішення базується на хмарних технологіях, перебої в інтернет-з’єднанні можуть впливати на стабільність роботи. Для компаній, які використовують традиційні підходи до управління, адаптація до нового ПЗ може стати викликом.

Попри високі стандарти безпеки, що застосовуються в Blue Yonder, ризики кіберзагроз, зокрема атак програм-вимагачів, залишаються актуальними, особливо для компаній, які працюють із великими обсягами конфіденційних даних.

Однак ці комерційні рішення мають ряд недоліків у контексті потреб компанії "Agrofusion".

Висока вартість впровадження та підтримки комерційних систем може зробити інвестицію економічно недоцільною для компанії, особливо якщо її потреби є вузькоспеціалізованими. Надлишковий функціонал у таких системах часто ускладнює користувацький інтерфейс та створює додаткові труднощі в процесі навчання персоналу.

Інтеграція з уже наявними системами компанії «Agrofusion» може бути складною та вимагати значних додаткових фінансових і часових ресурсів. Обмежена гнучкість комерційного програмного забезпечення щодо адаптації до специфічних бізнес-процесів компанії □зокрема сезонності виробництва та особливостей аграрного сектору □є ще одним суттєвим недоліком.

Залежність від постачальника програмного забезпечення при внесенні змін або адаптації системи до нових вимог бізнесу зменшує оперативність реагування на зміни та може стримувати розвиток компанії. Крім того, відсутність повної локалізації таких систем для українського ринку створює додаткові труднощі в роботі з бухгалтерськими та податковими системами, що є критично важливими для підприємства.

У зв’язку з вищезазначеними обмеженнями, розробка власного вебзастосунка для автоматизації внутрішніх перевезень є обґрунтованим кроком для компанії «Agrofusion». Такий підхід дозволяє повністю врахувати всі особливості бізнес-процесів компанії, включаючи сезонний характер виробництва та специфіку аграрної діяльності.

Створення власного рішення дає змогу оптимізувати витрати на розробку, впровадження та подальше обслуговування системи. Крім того, функціонал може впроваджуватись поступово, починаючи з найбільш критичних етапів логістики, що дозволяє уникнути перевантаження користувачів та забезпечити плавний перехід.

Власна розробка гарантує повний контроль над системою та її розвитком, дозволяє швидко адаптуватися до змін у бізнес-процесах і ефективно реагувати на нові виклики без зовнішніх залежностей.

1.3 Постановка задачі на розробку вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Виходячи з аналізу задачі інженерії програмного забезпечення, сформулюємо наступну постановку задачі: необхідно розробити вебзастосунок “Agrofusion”, який задовольняє наступним вимогам.

Основні функціональні вимоги:

– Створення заявок на перевезення, включаючи тип вантажу, обсяг, пункти відправлення та призначення.

Вхідні дані: тип вантажу, обсяг вантажу, пункт відправлення, пункт призначення, спеціальні умови для перевезення (температурний режим, вентиляція, вологість, великогабаритний вантаж), замовник.

– Погодження заявок відповідальною особою.

Вхідні дані: заява на вантажоперевезення, погоджувальний лист.

– Відстеження статусу заявок.

Вхідні дані: узгоджена заява на вантажоперевезення.

– Можливість друку необхідних документів, наприклад, товарно-транспортних накладних.

Вхідні дані: заява на вантажоперевезення.

– Зберігання історії статусів замовлення.

Вхідні дані: статус заяви на вантажоперевезення.

– Формування стандартних звітів про виконані перевезення.

Вхідні дані: заява на вантажоперевезення.

– Можливість створення користувацьких звітів та експорту даних у різні формати.

Вхідні дані: заяви на вантажоперевезення.

На основі введених вхідних даних вебзастосунок повинен генерувати наступні вихідні дані:

– Створення заявок на перевезення, включаючи тип вантажу, обсяг, пункти відправлення та призначення.

Вихідні дані: заява на вантажоперевезення.

– Погодження заявок відповідальною особою.

Вихідні дані: узгоджена заява на вантажоперевезення.

– Відстеження статусу заявок.

Вихідні дані: оновлений етап вантажоперевезення узгодженої заяви.

– Можливість друку необхідних документів, наприклад, товарно-транспортних накладних.

Вихідні дані: документ ТТН.

– Зберігання історії статусів замовлення.

Вихідні дані: журнал статусів для замовлення на вантажоперевезення.

– Можливість створення користувацьких звітів та експорту даних у різні формати.

Вихідні дані: Звіт про вантажоперевезення (залежно від потреб можна сформувати, наприклад за кількістю узгоджених або відхилених заяв).

Важливими аспектами також є безпека та продуктивність, оскільки вебзастосунок повинен стабільно працювати при високому навантаженні та гарантувати захист даних відповідно до встановлених стандартів. Інтерфейс користувача має бути інтуїтивно зрозумілим, адаптованим до потреб логістів і доступним для швидкого опанування.

2 РОЗРОБКА ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ЛОГІСТИЧНИХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ “AGROFUSION”

2.1 Аналіз вимог до вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Під час розробки програмного забезпечення важливо не лише створити функціональний продукт, а й ретельно спланувати його архітектуру та взаємодію з користувачем. Етап планування відіграє ключову роль у формуванні загального бачення системи, визначенні її основних компонентів та сценаріїв використання. Саме на цьому етапі розробник закладає фундамент для подальшої реалізації, тестування та вдосконалення програмного забезпечення.

Перший крок планування – це побудова діаграм Use Case та складання Use Case. Use Case (сценарій користування) – це перелік дій, сценарій, за яким користувач взаємодіє з додатком або програмою для виконання будь-якої дії та досягнення конкретної мети [6].

Для формалізації вимог до майбутнього програмного забезпечення була створена діаграма варіантів використання, яка відображає можливих акторів та зв’язки між ними. Така діаграма дає змогу проаналізувати структуру програмного забезпечення, основні функціональні можливості та способи взаємодії з зовнішніми сутностями.

На рисунку 2.1 представлено діаграму варіантів використання вебзастосунка для внутрішніх перевезень компанії "Agrofusion".

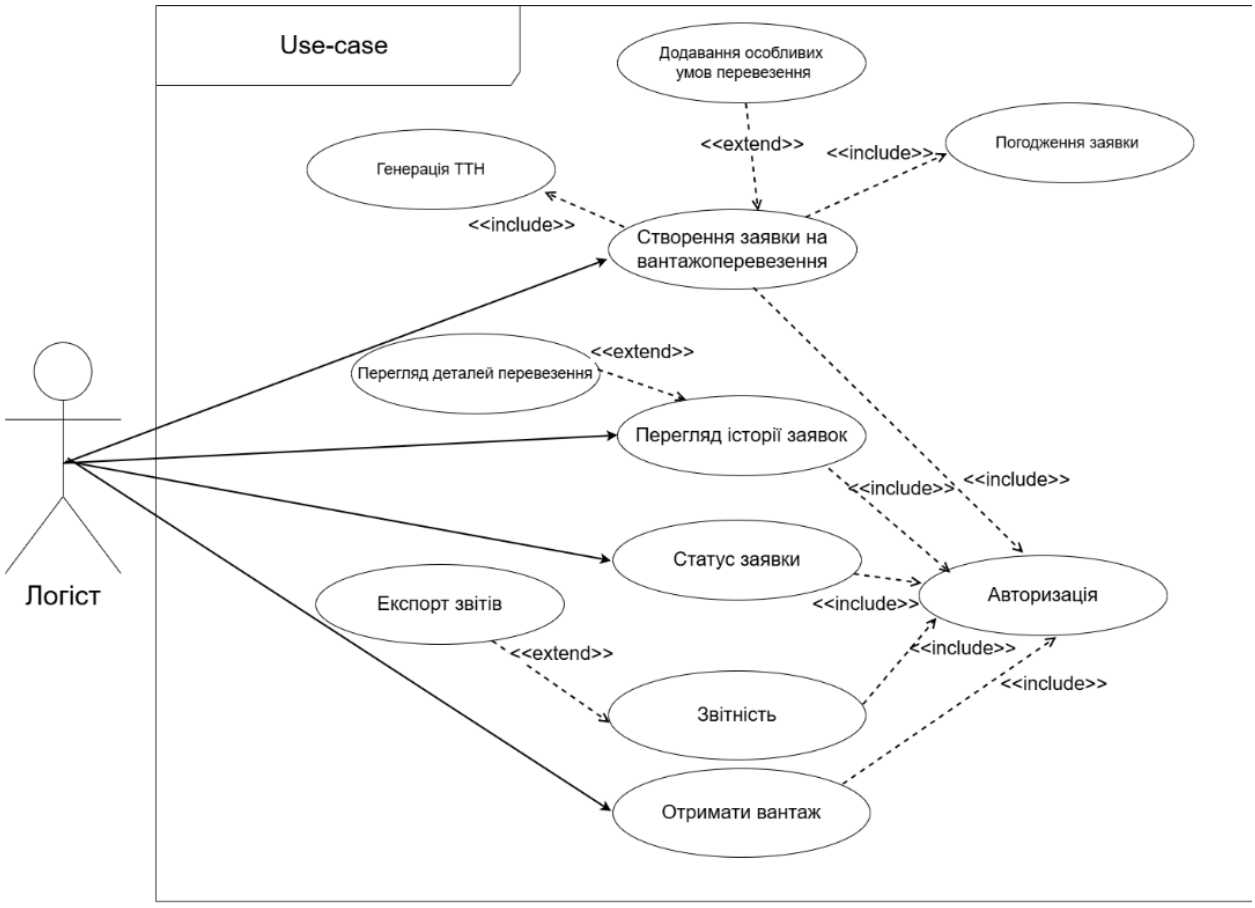


Рисунок 2.1 – Діаграма варіантів використання вебзастосунка для внутрішніх перевезень компанії "Agrofusion"

На основі проведеного аналізу діаграми варіантів використання сформовано специфікації для кожного окремого варіанту.

Специфікація варіанту використання «Створення заявки на вантажоперевезення».

1) Найменування: «Створення заявки на вантажоперевезення».

2) Призначення: для створення та редагування заявки на перевезення вантажу із зазначенням основних параметрів доставляння.

3) Учасники: Логіст.

4) Передумови: перед активізацією даного варіанта використання повинен бути виконаний потік варіанта використання «Авторизація».

5) Основний потік: користувач після авторизації переходить до форми створення заявки, вводить тип вантажу, обсяг, пункти відправлення та

призначення, спеціальні умови, після чого підтверджує заявку. Вебзастосунок зберігає її та надає відповідне підтвердження.

6) Альтернативні потоки: якщо не заповнено обов’язкові поля – відображається повідомлення про помилку, заявка не зберігається.

7) Спеціальні вимоги: валідація форми заявки (перевірка обсягу вантажу, коректності пунктів, обов’язковість полів); автоматичне збереження чернетки заявки при тривалому редагуванні.

8) Постумови: заявка створена та збережена в вебзастосунку, користувач може переглядати її статус.

Специфікація варіанту використання «Погодження заявки»

1) Найменування: «Погодження заявки»

2) Призначення: підтвердження створеної заявки відповідальною особою.

3) Учасники: Логіст.

4) Передумови: заявка повинна бути створена та збережена.

5) Основний потік: адміністратор переглядає заявку та натискає кнопку погодження. Статус заявки змінюється на «Погоджено».

6) Альтернативні потоки: відповідальна особа відхиляє заявку – статус змінюється на «Відхилено».

7) Спеціальні вимоги: вебзастосунок зберігає дату й час погодження, а також користувача, який його виконав.

8) Постумови: статус заявки змінений.

9) Додаткові зауваження: варіант використання розширює «Створення заявки».

Специфікація варіанту використання «Перегляд деталей перевезення»

1) Найменування: «Перегляд деталей перевезення».

2) Призначення: надати користувачу доступ до детальної інформації про перевезення.

3) Учасники: Логіст.

4) Передумови: користувач повинен бути авторизований.

5) Основний потік: користувач обирає заявку зі списку і переглядає її параметри: маршрут, умови, статус тощо.

6) Альтернативні потоки: якщо заявка недоступна – вебзастосунок повідомляє про помилку.

7) Спеціальні вимоги: має бути захищена від несанкціонованого доступу.

8) Постумови: дані переглянуто, заявка залишається незмінною.

9) Додаткові зауваження: відсутні.

Специфікація варіанту використання «Перегляд історії заявок»

1) Найменування: «Перегляд історії заявок».

2) Призначення: показати користувачу всі його попередні заявки.

3) Учасники: Логіст.

4) Передумови: користувач авторизований.

5) Основний потік: користувач відкриває розділ «Історія заявок», бачить список зі статусами й датами.

6) Альтернативні потоки: якщо історія пуста – виводиться відповідне повідомлення.

7) Спеціальні вимоги: можливість фільтрації за періодом або статусом.

8) Постумови: дані переглянуто.

9) Додаткові зауваження: відсутні.

Специфікація варіанту використання «Формування та перегляд звітів»

1) Найменування: «Перегляд статусу заявок».

2) Призначення: створення звітів на основі заявок.

3) Учасники: Логіст.

4) Передумови: у системі наявні дані.

5) Основний потік: користувач обирає період/тип звіту, вебзастосунок формує та відображає звіт.

6) Альтернативні потоки: якщо даних немає – виводиться повідомлення.

7) Спеціальні вимоги: можливість фільтрації, сортування, форматування звітів.

8) Постумови: звіт сформовано.

9) Додаткові зауваження: розширюється варіантом <<extend>> «Експорт звітів».

2.2 Розробка архітектури вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Розробка коду починається з побудови архітектури програми й від її якості залежить не стільки поточне функціонування програми, а її модифікація в майбутньому. Зазвичай, не виникає проблем з архітектурою програми відразу після створення продукту і, якщо ПЗ є сталим, проблеми можуть і не проявлятися. Але в сучасному динамічному світі все дуже швидко розвивається та змінюється [7].

Серед можливих варіантів розглядалися монолітна, мікросервісна, REST API та MVC-архітектури. Моноліт проста у реалізації, але важка в масштабуванні. Мікросервіси забезпечують гнучкість, але потребують складнішого керування. REST API полегшує інтеграцію, хоча має обмеження. Архітектура MVC (Model-View-Controller) вирізняється чітким розділенням логіки: модель відповідає за дані, подання – за інтерфейс, а контролер – за обробку запитів і зв’язок між компонентами.

MVC обрана як найкращий варіант для цього проєкту, оскільки вона забезпечує гнучкість, масштабованість і структурованість. Вебзастосунок автоматизує управління заявками на перевезення, їх погодження, відстеження та генерацію супровідної документації. Архітектура MVC повністю відповідає технічним вимогам до функціональності, надійності та безпеки.

На рисунку 2.2 представлено діаграму компонентів вебзастосунка для внутрішніх перевезень компанії "Agrofusion".

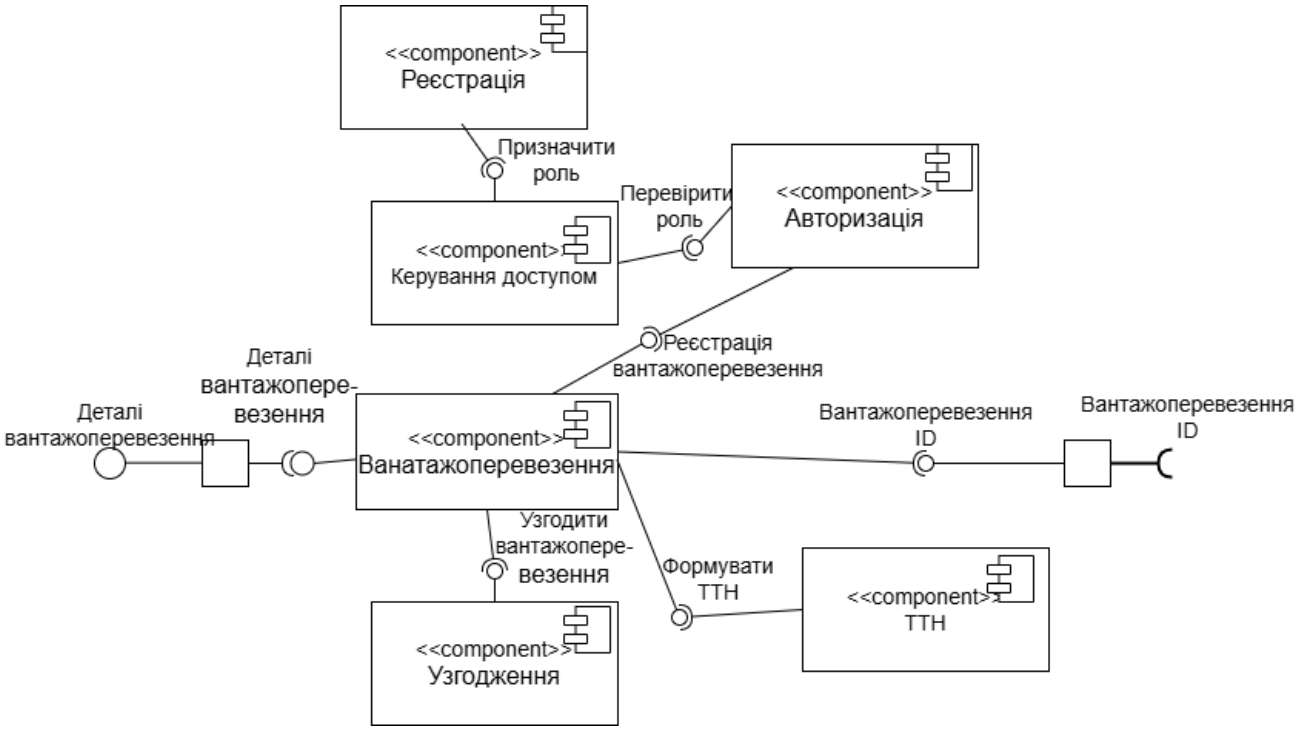


Рисунок 2.2 – Діаграма компонентів вебзастосунка для внутрішніх перевезень компанії "Agrofusion"

Вебзастосунок складається з кількох компонентів, які взаємодіють між собою через інтерфейси. Компонент “Реєстрація” здійснює реєстрацію нового користувача, і пов’язаний із компонентом “Керування доступом”. Компонент “Керування доступом” відповідає за призначення і контроль прав користувача і, своєю чергою, взаємодіє з компонентом “Авторизація”, який забезпечує автентифікацію.

Компонент “Вантажоперевезення” є центральним у вебзастосунку, він відповідає за обробку вантажоперевезень. Він має інтерфейси для отримання та надання даних про деталі вантажоперевезення, а також інтерфейс для передачі ідентифікатора вантажоперевезення (ID) назовні. Крім того, компонент “Вантажоперевезення” пов’язаний із компонентом “Узгодження”, через інтерфейс “Узгодити вантажоперевезення”, і з компонентом “ТТН”, через інтерфейс “Формувати ТТН”. Компонент “Узгодження” використовується для погодження перевезення, а компонент “ТТН” – для формування товарно-транспортної накладної. Таким чином, вебзастосунок забезпечує повний цикл роботи з

вантажоперевезеннями – від реєстрації й авторизації користувача до погодження перевезення і формування ТТН.

На рисунку 2.3 представлено діаграму розгортання вебзастосунка для внутрішніх перевезень компанії "Agrofusion".

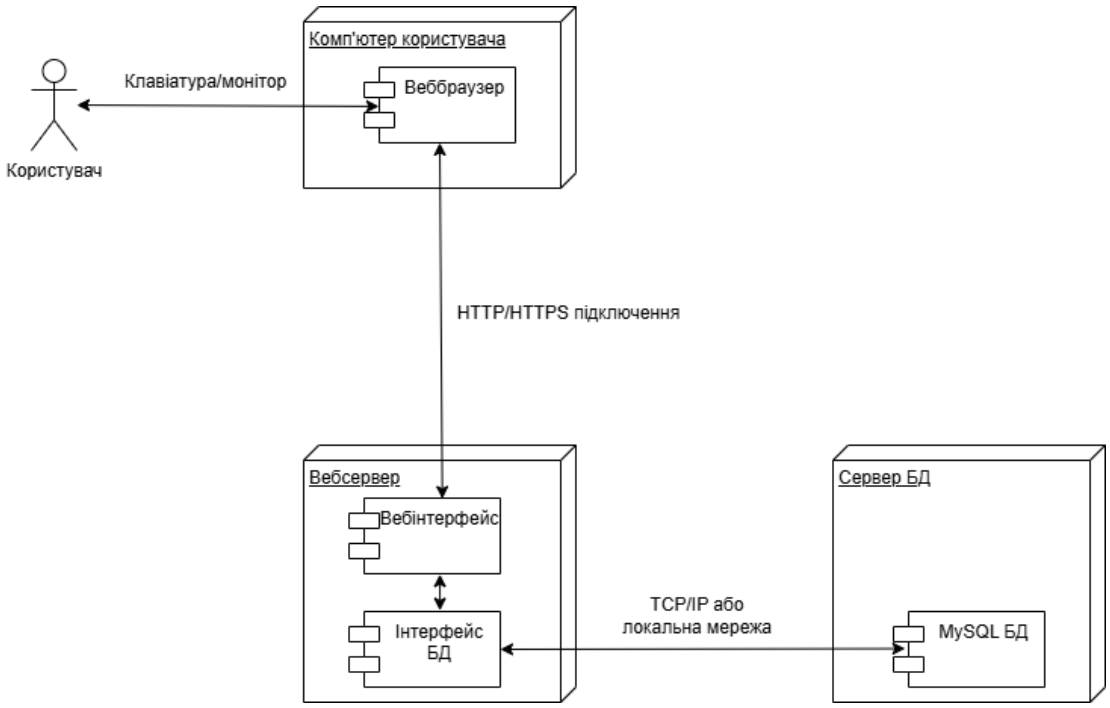


Рисунок 2.3 – Діаграма розгортання вебзастосунка для внутрішніх перевезень компанії "Agrofusion"

На діаграмі зображено архітектуру трирівневого вебзастосунку, що включає користувача, веббраузер, вебсервер та сервер бази даних. Користувач взаємодіє із вебзастосунком за допомогою клавіатури та монітора через веббраузер, який працює на комп’ютері користувача. Веббраузер встановлює HTTP або HTTPS з’єднання з вебсервером. На вебсервері функціонують два основні компоненти: вебінтерфейс, який відповідає за обробку запитів користувача, та інтерфейс бази даних, що забезпечує доступ до сервера бази даних. Вебінтерфейс отримує дані від користувача, обробляє їх і за потреби звертається до бази даних через інтерфейс БД. З’єднання між вебсервером і сервером бази даних здійснюється через TCP/IP протокол або локальну мережу. На сервері бази даних розміщено компонент

MySQL БД, який зберігає та надає дані для вебсервера. Таким чином, користувач отримує доступ до інформаційної системи через веббраузер, а обробка даних та їх зберігання відбувається на віддалених серверах.

У контексті архітектури MVC, об'єктноорієнтована методологія дозволяє природно структурувати код і логіку додатка. Компонент Model реалізується у вигляді класів, що представляють бізнес-логіку та дані – це можуть бути сутності, сервіси або модулі роботи з базами даних. Компонент View реалізується через класи або шаблони, які відповідають за представлення інформації користувачеві (інтерфейс), і може включати механізми відображення даних, відгуку на дії користувача, а також елементи стилізації. Компонент Controller втілюється як окремі об'єкти або класи, які отримують запити від View, обробляють їх (часто взаємодіючи з Model), і визначають, які View оновити в результаті.

2.3 Розробка проєкту вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Сучасні розробники мають широкий вибір технологій для створення інформаційних систем, однак ефективність розробки залишається низькою. Згідно з даними, до 40% проєктів не завершуються, понад 70% – не досягають цілей, а строки реалізації в середньому перевищуються на 220%. Часто кінцевий результат не відповідає вимогам, а замовники недостатньо залучені до процесу, що впливає на якість продукту [7].

Вибір методології розробки ПЗ є критичним, адже визначає темп, складність і структуру проєкту. Методологію слід підбирати з урахуванням завдань і обраної архітектури, яка вже визначена.

Об'єктноорієнтована методологія (ООМ) є одним із найпоширеніших підходів, що добре поєднується з архітектурою MVC. В її основі – моделювання системи як набору об'єктів, які відображають сутності предметної області та взаємодіють через інтерфейси.

Діаграма діяльності є графічним інструментом для моделювання процесів, зокрема процесу прийняття рішень. Вона дозволяє наочно зобразити послідовність дій, що виконуються під час цього процесу, а також показати потік інформації та взаємодію між окремими етапами. Основними компонентами діаграми є: стани – дії або види діяльності; перехідні зв’язки (робота) – стрілки, що відображають логічні зв’язки між діями; а також рішення – елементи, які вказують на наявність вибору чи альтернативного шляху розвитку подій [8]. Діаграма сутності “Заява на вантажоперевезення” представлена на рисунку 2.4.

Користувач вебзастосунка проходить процедуру авторизації. Якщо користувач не зареєстрований або не правильно вводить логін і пароль – користувач отримує помилку, після чого повертається до авторизації. У разі успішної авторизації користувач переходить на сторінку «Оформити вантажоперевезення», де створює заявку.

Після створення заявки відбувається перевірка валідності даних. Якщо дані не відповідають вимогам, користувач повинен їх виправити до валідної форми. У випадку правильних даних формується лист узгодження.

Далі запускається процедура узгодження. Якщо заявку не погоджено – користувач вказує причину відхилення, і їй присвоюється статус «Відхилено». Якщо погоджено – присвоюється статус «На завантаження».

Діаграма діяльності сутності «Звіт з вантажоперевезення» представлена на рисунку 2.5.

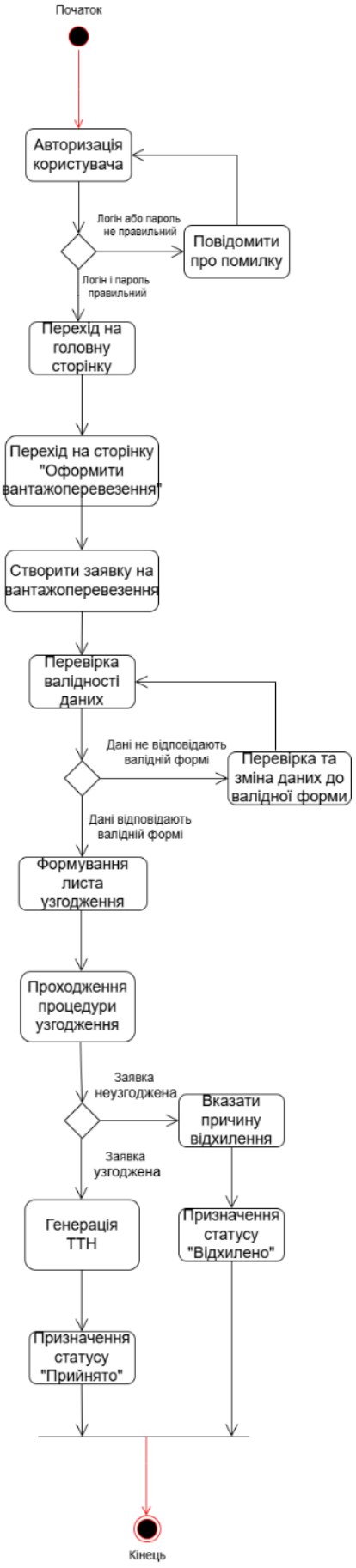


Рисунок 2.4 – Діаграма діяльності “Заява на вантажоперевезення”

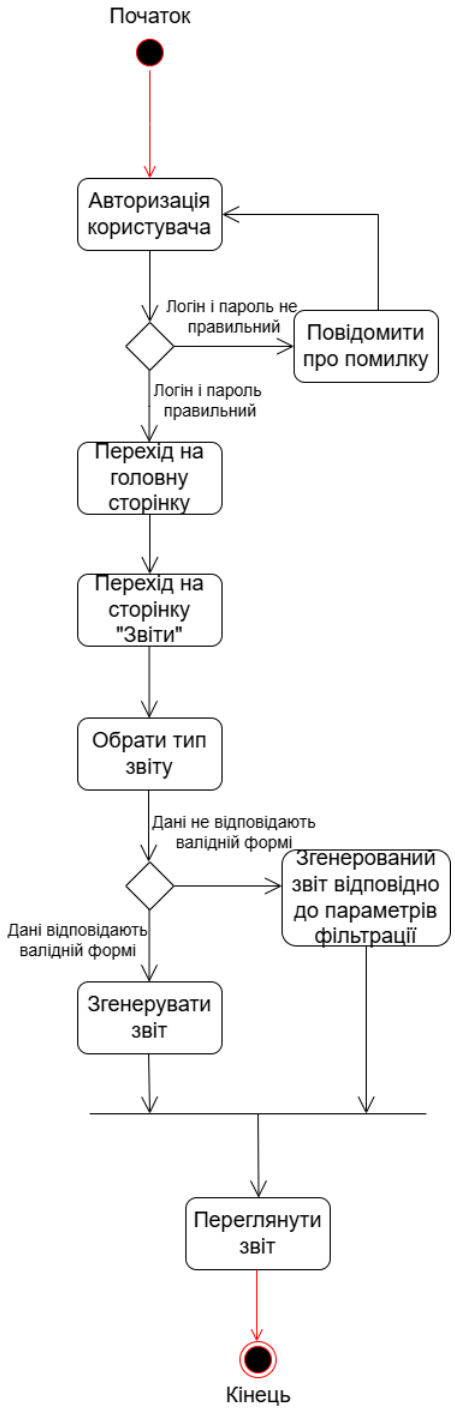


Рисунок 2.5 – Діаграма діяльності «Звіт з вантажоперевезення»

Користувач вебзастосунка проходить процедуру авторизації. Якщо користувач не зареєстрований або не правильно вводить логін і пароль – користувач отримує помилку, після чого повертається до авторизації. Якщо авторизація успішна відбувається перехід на сторінку "Звіти". На цій сторінці користувач має можливість обрати тип звіту, який він хоче згенерувати.

Після вибору типу звіту, користувач може вирішити, чи бажає він вказати параметри фільтрації для звіту. Якщо користувач вирішує застосувати фільтрацію (шлях "Так"), вебзастосунок генерує звіт відповідно до обраних параметрів фільтрації.

Якщо користувач не бажає встановлювати параметри фільтрації (шлях "Ні"), він переходить безпосередньо до генерування звіту зі стандартними параметрами.

Незалежно від обраного шляху, після генерування звіту користувач переходить до етапу "Переглянути звіт", де може ознайомитися з результатами. Після цього процес завершується і досягає позначки "Кінець".

При проєктуванні об'єктноорієнтованих програмних застосунків діаграми класів є обов’язковими. Діаграма класів – це набір статичних, декларативних елементів моделі. Слід зазначити, що діаграми класів можуть застосовуватися і при прямому (в процесі розробки нового вебзастосунка), і при зворотному проєктуванню – описі наявних інформаційних систем [9].

У процесі розробки вебзастосунку було створено діаграму класів, яка відображає основну структуру даних та взаємозв’язки між об'єктами предметної області.

На рисунку 2.6 представлено діаграму класів модуля «Користувач».

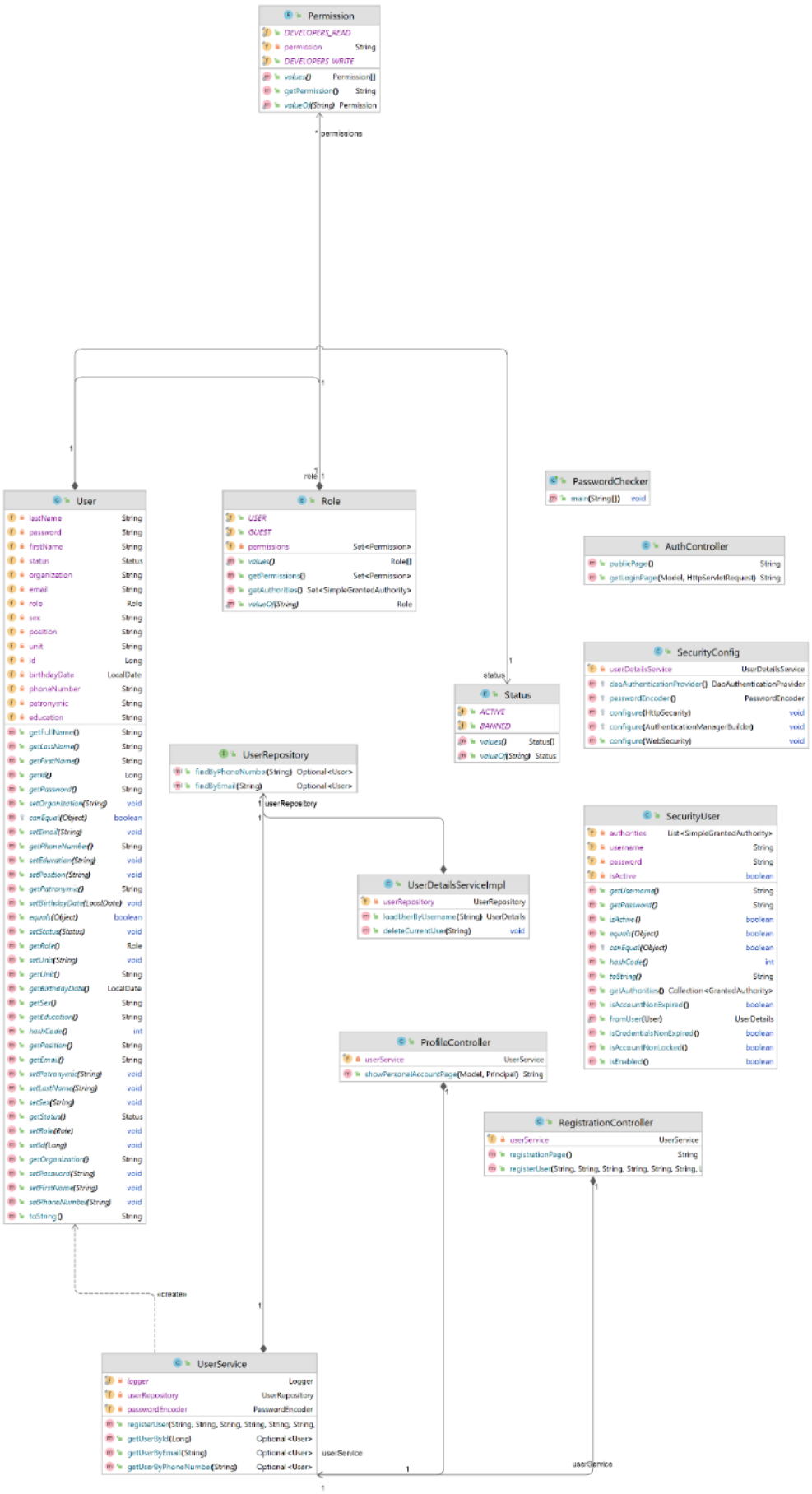


Рисунок 2.6 – Діаграма класів модуля «Користувач»

Специфікація класу «SecurityConfig»

Ім'я: SecurityConfig.

Відповідальність: налаштування безпеки вебзастосунка, включаючи автентифікацію, авторизацію, шифрування паролів, роботу з UserDetailsService і захист доступу до ресурсів.

Атрибути: userDetailsService – сервіс, який забезпечує завантаження користувачів для автентифікації, реалізований через customUserDetailsServiceImpl.

Операції зі специфікаціями нетривіальних операцій: configure(HttpSecurity http) – метод, що конфігурує правила доступу до HTTP-запитів. В результаті: доступ до кореневої сторінки (/), сторінки реєстрації (/registration), API (/api/**) та деяких сервісів (/service/receive-cargo) дозволено всім користувачам; усі інші запити вимагають автентифікації; налаштовується сторінка входу (/auth/login), сторінка після успішного входу (/auth/index), обробка помилки входу (/auth/login?error=true) та вихід із системи (/auth/logout).

configure(AuthenticationManagerBuilder auth) – метод, що конфігурує менеджер автентифікації. В результаті: підключається userDetailsService для завантаження даних користувача; використовується BCryptPasswordEncoder із силою шифрування 12 для перевірки паролів.

configure(WebSecurity web) – метод, який виключає деякі шляхи з безпекових обмежень. В результаті: шляхи /static/**, /images/**, /css/**, /js/**, /json/** ігноруються системою безпеки (доступні без авторизації).

passwordEncoder() – метод, який створює екземпляр об’єкта для шифрування паролів. В результаті: повертається BCryptPasswordEncoder з силою шифрування 12.

daoAuthenticationProvider() – метод, який конфігурує DAO-постачальника автентифікації. В результаті: повертається об’єкт DaoAuthenticationProvider, який використовує userDetailsService для отримання користувачів та passwordEncoder() для перевірки паролів.

Специфікація класу «ApiController»

Ім’я: ApiController.

Відповідальність: контролер REST API, що обробляє запити на отримання інформації про користувачів, адреси, вантажі, замовлення, історію змін статусів, а також дозволяє оновлення статусу замовлень.

Атрибути:

userService – сервіс для доступу до інформації про користувачів;

addressesService – сервіс для роботи з адресами;

orderService – сервіс для обробки замовлень;

technicCharacterCargoService – сервіс для доступу до характеристик вантажу;

logger – об'єкт для логування подій контролера;

getSender(Long id) – отримує користувача-відправника за ID. Якщо знайдено успішно повертає User, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getReceiver(Long id) – отримує користувача-отримувача за ID. Якщо знайдено успішно повертає User, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getAddress(Long id) – повертає адресу за ID. Якщо знайдено успішно повертає Address, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getCargo(Long id) – отримує інформацію про вантаж за ID. Якщо знайдено успішно повертає TechnicCharacterCargo, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getStatusHistory(Long orderId) – повертає історію змін статусів замовлення. Якщо знайдено успішно повертає список об'єктів OrderStatusHistoryDto із orderId, status, changedAt, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getUserOrders(Long userId, StatusUpdateRequest request) – повертає список замовлень користувача. отримує інформацію про вантаж за ID. Якщо знайдено успішно повертає Order, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

getUserOrdersLastStatusDate(Long userId) – Повертає дату останньої зміни статусу кожного замовлення користувача. Якщо знайдено успішно повертає OrderLastStatusDateDto, не знайдено, повертає статус 404. При помилці сервера повертає статус 500.

updateOrderStatus(Long orderId, StatusUpdateRequest request) – Оновлює статус замовлення. Якщо оновлення проходить успішно, змінює статус і повідомляє про його оновлення не оновлює, повертає статус 404. При помилці сервера повертає статус 500.

Специфікація класу «ApprovalController»

Ім’я: ApprovalController.

Відповідальність: контролер для взаємодії з листом узгодження (approval sheet) щодо замовлення – перегляд, коментування та підтвердження/відхилення.

Атрибути:

orderService – сервіс для доступу до замовлень;

userRepository – доступ до поточного користувача за email;

approverRepository – CRUD-операції з узгоджувачами (Approver);

approvalSheetService – логіка для роботи з листами узгодження (ApprovalSheet), включно з перевіркою прав;

Операції: getApprovalPage(Long orderId, Model model, Principal principal) – метод GET /approvals/{orderId}. Призначення: відображення сторінки узгодження для конкретного замовлення. Отримує замовлення за orderId. Визначає поточного користувача через Principal. Перевіряє, чи має користувач право на підтвердження цього замовлення (isApprover(...)). Якщо користувач є узгоджувачем, відображає сторінку "approval-page" із даними order, user, approvalSheet. У разі помилки – сторінка "error" з відповідним повідомленням.

handleApprovalAction(Long orderId, Long approverId, String action, String comment, Principal principal) – метод POST /approvals/action. Призначення: обробка дій узгоджувача (коментар, схвалення, відхилення). Отримує об’єкт Approver за approverId. Якщо action відсутній або дорівнює "COMMENT": Зберігає коментар

без зміни статусу. Якщо статус вже не PENDING , не дозволяє змінити рішення повторно. Інакше оновлює статус відповідно до APPROVED або REJECTED, зберігає коментар та перевіряє, чи всі узгоджувачі вже схвалили (APPROVED). Якщо так – викликає orderService.finalizeOrder(...).

Специфікація класу «AuthController»

Ім’я: AuthController.

Відповідальність: обробка запитів, пов’язаних із автентифікацією користувачів – зокрема, відображення сторінки логіну та публічної домашньої сторінки.

Операції: getLoginPage(Model model, HttpServletRequest request) – повертає сторінку авторизації. Перевіряє наявність параметра "error" у URL (через request.getQueryString()). Повертає ім’я представлення: "login".

publicPage() – повертає публічну домашню сторінку (наприклад, загальнодоступну для всіх).

Специфікація класу «ReceiverController»

Ім’я: ReceiverController

Відповідальність: за обробку запитів, пов’язаних з отриманням вантажів користувачем, переглядом історії завершених замовлень і переглядом історії змін статусів замовлень.

Атрибути:

orderService – сервіс для взаємодії з об'єктами замовлень (Order);

userService – сервіс для отримання даних автентифікованого користувача;

historyService – сервіс для збереження інформації про завершені замовлення;

addressesService – сервіс для обробки адрес замовлень (не використовується у поточному коді, але підключений);

Операції зі специфікаціями нетривіальних операцій:

getMyOrders(Model model, Principal principal) – перевіряється автентифікація користувача. Отримується список замовлень, де поточний користувач є одержувачем.

confirmOrder(@RequestBodyOrderConfirmationData orderConfirmationData) – отримується orderId із переданих клітинок таблиці. Замовлення знаходиться по orderId. Замовлення оновлюється, зберігається, і записується в історію. Повертається відповідь 200 OK або помилка 400 Bad Request.

getCompletedOrders(Model model, Principal principal) Викликається при GET-запиті на /service/history-cargo. У результаті запуску методу отримуються завершені замовлення (зі статусом DONE) для поточного користувача.

getStatusHistory(Model model, Principal principal) – у результаті запуску методу отримується історія останніх змін статусів замовлень для користувача.

Специфікація класу «SendCargoController»

Ім'я: SendCargoController.

Відповідальність: обробка HTTP-запитів, пов’язаних із створенням та надсиланням замовлень на вантажні перевезення. Забезпечує взаємодію між фронтендом (формами) та сервісним шаром для створення заявки, реєстрації адрес і характеристик вантажу, а також ініціалізації процесу узгодження.

Атрибути:

TechnicCharacterCargoService – сервіс для роботи з характеристиками вантажу.

AddressesService addressesService – сервіс для роботи з адресами доставлення.

OrderService orderService – сервіс для роботи з замовленнями.

UserRepository userRepository – репозиторій для доступу до інформації про користувачів.

ApprovalSheetService approvalSheetService – сервіс для управління процесом узгодження замовлень.

Logger logger – логер для ведення журналу подій і помилок.

Операції зі специфікаціями нетривіальних операцій:

String getSendCargoPage(Model model, Principal principal) – Метод, що обробляє GET-запит на сторінку надсилання вантажу.

String sendCargo(OrderForm orderForm, Principal principal, RedirectAttributes redirectAttributes, CargoForm cargoForm) – Метод, що обробляє POST-запит при відправленні форми створення замовлення.

Специфікація класу «CargoForm»

Ім'я: CargoForm.

Відповідальність: служить формою для передачі даних про вантаж та адресу доставлення між вебшаром (контролером) і користувацьким інтерфейсом. Забезпечує інкапсуляцію характеристик вантажу та адреси для подальшої обробки у бізнес-логіці.

Атрибути:

TechnicCharacterCargo cargo – об’єкт, що містить характеристики вантажу (вага, тип, умови зберігання тощо).

Addresses addresses – об’єкт, що містить інформацію про адресу відправлення, проміжні точки та адресу доставлення.

Операції зі специфікаціями нетривіальних операцій: CargoForm() – конструктор за замовчуванням, що ініціалізує поля cargo і addresses новими порожніми об’єктами відповідних класів.

Специфікація класу «OrderForm»

Ім'я: OrderForm.

Відповідальність: форма для передачі даних замовлення вантажу між шаром контролера і користувацьким інтерфейсом. Інкапсулює інформацію про замовлення, таку як адреса, вантаж, відправник і отримувач, а також деталі доставлення (дата, час тощо).

Атрибути:

Long id – унікальний ідентифікатор замовлення.

Long addressId – ідентифікатор адреси доставлення.

Long cargoId – ідентифікатор вантажу.

Long clientSenderId – ідентифікатор користувача-відправника.

String receiverPhoneNumber – номер телефону отримувача.

Long clientReceiverId – ідентифікатор користувача-отримувача.

LocalDate orderDate – дата створення замовлення.

LocalTime orderTime – час створення замовлення.

boolean isDelivered – статус доставлення (чи було доставлено).

Операції зі специфікаціями нетривіальних операцій:

OrderForm() – конструктор за замовчуванням, ініціалізує об'єкт без параметрів.

String toString() – метод для представлення об'єкта у вигляді рядка з усіма ключовими атрибутами замовлення, що зручно для логування і налагодження.

Специфікація класу «Addresses»

Ім'я: Addresses.

Відповідальність: представляє адресу доставлення в вебзастосунку. Зберігає інформацію про початкову точку відвантаження, до трьох проміжних точок і кінцеву точку доставлення. Використовується для збереження адресної інформації у базі даних.

Атрибути:

Long id – унікальний ідентифікатор адреси (генерується базою даних).

String shippingPoint – адреса або місце відправлення вантажу.

String intermediatePoints1 – перша проміжна точка маршруту (може бути порожньою).

String intermediatePoints2 – друга проміжна точка маршруту (може бути порожньою).

String intermediatePoints3 – третя проміжна точка маршруту (може бути порожньою).

String deliveryPoint – кінцева точка доставлення вантажу.

Операції зі специфікаціями нетривіальних операцій: клас містить гетери і сетери (згенеровані Lombok анотацією @Data). Використовується JPA-Entity для збереження у таблиці addresses бази даних з автоматичною генерацією ID.

Специфікація класу «ApprovalSheet»

Ім'я: ApprovalSheet.

Відповідальність: зберігання аркуша узгодження для замовлення, який включає статус узгодження та список узгоджувачів.

Атрибути:

order – Замовлення, для якого створений аркуш узгодження.

status – Загальний статус погодження (PENDING, APPROVED тощо).

approvers – Список узгоджувачів(користувачів), які мають погодити замовлення.

id – Унікальний ідентифікатор аркуша узгодження.

Операції зі специфікаціями нетривіальних операцій:

getOrderId() – цей метод повертає ідентифікатор замовлення, пов’язаного з аркушем узгодження. У результаті запуску даного методу: повертається null, якщо замовлення не прив'язане, інакше – order.getId().

isFullyAgreed() – цей метод перевіряє, чи всі узгоджувачі зі списку погодили замовлення. У результаті запуску даного методу повертається true, якщо список узгоджувачів не порожній і всі вони мають статус APPROVED; інакше – false.

Специфікація класу «Approver»

Ім’я: Approver.

Відповідальність: збереження інформації про узгоджувача (користувача), який бере участь в узгодженні замовлення, його статус погодження та коментар.

Атрибути:

id – унікальний ідентифікатор узгоджувача.

status – статус погодження з боку конкретного користувача (PENDING, APPROVED, REJECTED).

comment – коментар, залишений узгоджувачем щодо погодження.

approvalSheet – посилання на аркуш узгодження, до якого належить погоджувач.

user – користувач, який є узгоджувачем.

Операції зі специфікаціями нетривіальних операцій: відсутні складні методи. Усі операції автоматично обробляються за допомогою Lombok (@Data) та JPA.

Специфікація класу «Order»

Ім’я: Order.

Відповідальність: представлення замовлення, включаючи відомості про відправника, отримувача, адресу, вантаж, статус замовлення та історію змін статусів.

Атрибути:

id – Унікальний ідентифікатор замовлення.

orderDate – Дата створення замовлення.

orderTime – Час створення замовлення.

clientSender – Об’єкт типу User, який є відправником вантажу.

clientReceiver – Об’єкт типу User, який є отримувачем вантажу.

address – Об’єкт типу Addresses, що містить маршрут доставлення.

cargo – Об’єкт типу TechnicCharacterCargo, що описує характеристики вантажу.

approvalSheet – Аркуш погодження (ApprovalSheet), пов’язаний із цим замовленням.

status – Поточний статус замовлення (OrderStatus). За замовчуванням DRAFT.

statusHistory – Список змін статусів замовлення (OrderStatusHistory).

lastUpdated – Дата та час останнього оновлення запису про замовлення.

Операції зі специфікаціями нетривіальних операцій: явно не визначено додаткових методів, однак завдяки анотації @Data (Lombok) автоматично створюються гетери/сетери для всіх полів toString(), equals(), hashCode()

Специфікація класу «OrderStatusHistory»

Ім’я: OrderStatusHistory.

Відповідальність: відображає історію змін статусів замовлення. Зберігає статус, час зміни та зв’язок із відповідним замовленням. Використовується для аудиту та відображення історії замовлення.

Атрибути:

Long – Унікальний ідентифікатор запису історії.

OrderStatus – Статус замовлення (DRAFT, APPROVED, DELIVERED).

LocalDateTime – Дата і час зміни статусу.

Order – Замовлення, до якого належить запис зміни статусу.

Операції зі специфікаціями нетривіальних операцій: методи, згенеровані анотацією @Data (Lombok) гетери/сетери, equals(), hashCode(), toString()

Специфікація класу «User»

Ім’я: User.

Відповідальність: моделює клієнта/користувача вебзастосунку із персональними, контактними, освітніми та організаційними даними, а також роллю й статусом.

Атрибути:

Id – Унікальний ідентифікатор користувача

PhoneNumber – Номер телефону користувача

FirstName – Ім’я

LastName – Прізвище

Patronymic – По батькові

BirthdayDate – Дата народження

Sex – Стать

Education – Освіта

Organization – Назва організації, де працює користувач

Unit – Підрозділ (наприклад, відділ логістики)

Position – Посада користувача

Email – Електронна адреса

Password – Пароль користувача (зашифрований)

Status – Поточний статус користувача (активний, заблокований тощо)

Role – Роль користувача в вебзастосунку (CLIENT, ADMIN, APPROVER тощо)

Операції зі специфікаціями нетривіальних операцій: GetFullName() – Повертає повне ім’я користувача у форматі: Ім’я Прізвище По батькові. Гетери / Сетери (згенеровані) Equals(), HashCode(), ToString()

Специфікація Enum «Role»

Ім’я: Role.

Відповідальність: визначає ролі користувачів системи і їхні права доступу.

Атрибути: Permissions – Набір прав доступу (Set<Permission>)

Методи: getPermissions() – повертає набір прав доступу ролі; getAuthorities() – повертає набір Spring Security SimpleGrantedAuthority на основі прав доступу.

Специфікація класу «PasswordChecker»

Ім’я: PasswordChecker.

Відповідальність: створює простий HTTP-сервер, який слухає вхідні POST-запити на перевірку пароля. Сервер обробляє запити на /check-password і відповідає JSON-рядком з інформацією про те, чи правильний пароль.

Атрибути: відсутні (усі поля локальні або всередині методу).

Операції зі специфікаціями нетривіальних операцій: main(String[] args) – Запускає HTTP-сервер на вказаному порту (3306). Створює контекст /check-password, в якому обробляються POST-запити.

Специфікація класу «SecurityUser»

Ім’я: SecurityUser.

Відповідальність: є адаптером між сутністю User та механізмом автентифікації Spring Security. Реалізує інтерфейс UserDetails, надаючи фреймворку інформацію про ім’я користувача, пароль, ролі (authorities) й активність облікового запису.

Атрибути:

Username – Логін користувача (e-mail).

Password – Пароль користувача (у зашифрованому вигляді).

Authorities – Список повноважень (List<SimpleGrantedAuthority>), що відповідають ролі користувача.

IsActive – Прапорець, який відображає активність облікового запису (true – активний).

Специфікація класу «UserService»

Назва: UserService.

Призначення: надає API для доступу, реєстрації та пошуку користувачів у системі. Використовується як допоміжний сервіс безпеки та для реєстрації нових користувачів.

Атрибути:

UserRepository – Репозиторій для доступу до даних користувачів.

PasswordEncoder – Компонент для хешування паролів перед збереженням.

Logger – Логер для ведення діагностичних повідомлень, зокрема при пошуку користувача за номером телефону.

Операції зі специфікаціями нетривіальних операцій:

Optional<User> getUserById(Long id) – Повертає користувача за його ID.

Optional<User> getUserByEmail(String email) – Повертає користувача за його email.

Optional<User> getUserByPhoneNumber(String phoneNumber) – Повертає користувача за номером телефону.Додатково логуються повідомлення про успіх або невдачу пошуку.

User registerUser(...) – Реєструє нового користувача за наданими даними.

Специфікація інтерфейсу «AddressesService»

Ім'я: AddressesService.

Відповідальність: інтерфейс визначає набір операцій для роботи з адресами доставлення, зокрема для отримання адреси за унікальним ідентифікатором та реєстрації нових адресних даних.

Атрибути: Інтерфейс не містить атрибутів, оскільки визначає тільки контракт (методи).

Операції зі специфікаціями нетривіальних операцій: Optional<Addresses> getAddressById(Long id) – спрацьовує при необхідності отримати адресу за її унікальним ідентифікатором.

Специфікація класу «AddressesServiceImpl»

Ім'я: AddressesServiceImpl

Відповідальність: реалізує інтерфейс AddressesService, забезпечуючи роботу з адресами доставлення – отримання за ID та реєстрацію нових адресних записів у базі даних.

Атрибути: addressesRepository – репозиторій для доступу до сутності Addresses, виконує операції збереження та пошуку адрес у базі.

Операції зі специфікаціями нетривіальних операцій: Optional<Addresses> getAddressById(Long id) – Метод виконує пошук адреси у базі за унікальним ідентифікатором; Addresses registerAddresses(String shippingPoint, String intermediatePoints1, String intermediatePoints2, String intermediatePoints3, String deliveryPoint) – створює новий запис адреси з усіма вказаними пунктами маршруту та зберігає його у базі даних.

Специфікація інтерфейсу «ApprovalSheetService»

Ім'я: ApprovalSheetService

Відповідальність: Забезпечує операції для роботи з листом узгодження (ApprovalSheet), включаючи створення початкового листа, перевірку ролі узгоджувача та пошук листа за ID замовлення.

Операції зі специфікаціями нетривіальних операцій:

ApprovalSheet createInitialApprovalSheet(Long orderId, List<Long> approverIds) – Метод створює початковий лист узгодження для заданого замовлення з вказаними ID узгоджувачів.

boolean isUserApprover(Long orderId, Long userId) – Метод перевіряє, чи є користувач з вказаним ID узгоджувачем для конкретного замовлення.

Optional<ApprovalSheet> findByOrderId(Long orderId) – Метод шукає лист узгодження за ідентифікатором замовлення.

Специфікація класу «ApprovalSheetServiceImpl»

Ім'я: ApprovalSheetServiceImpl.

Відповідальність: реалізує бізнес-логіку для роботи з листом узгодження (ApprovalSheet), включаючи створення початкового листа узгодження, перевірку ролі узгоджувача для конкретного замовлення, та пошук листа за ID замовлення.

Атрибути:

approvalSheetRepository – репозиторій для збереження та отримання ApprovalSheet.

approverRepository – репозиторій для роботи з узгоджувачами (Approver).

orderRepository – репозиторій для доступу до замовлень (Order).

userRepository – репозиторій для доступу до користувачів (User).

Операції зі специфікаціями нетривіальних операцій: ApprovalSheet createInitialApprovalSheet(Long orderId, List<Long> approverIds) – створює початковий лист узгодження для замовлення з ідентифікатором orderId та списком узгоджувачів за їх ID.

boolean isUserApprover(Long orderId, Long userId) – перевіряє, чи є користувач з userId узгоджувачем листа узгодження для замовлення з orderId.

Optional<ApprovalSheet> findByOrderId(Long orderId) – пошук ApprovalSheet за ідентифікатором замовлення.

Специфікація класу «HistoryService»

Ім'я: HistoryService.

Відповідальність: збереження історичних записів про замовлення в базу даних для подальшого аудиту або перегляду.

Атрибути: historyRepository – репозиторій для роботи з сутністю HistoryRecord, що забезпечує збереження й отримання історичних даних.

Операції зі специфікаціями нетривіальних операцій:

void saveToHistory(Order order) – створення нового запису історії на основі об'єкта замовлення (Order) та збереження його в базу даних.

Специфікація інтерфейсу «OrderService»

Ім'я: OrderService.

Відповідальність: управління замовленнями, їхнім статусом, історією змін та отриманням інформації по замовленнях за різними критеріями.

Операції:

Long saveDraftOrder(OrderForm orderForm) – Зберігає замовлення у статусі чернетки.

Optional<Order> getOrderById(String orderId) – Повертає замовлення за його ідентифікатором у вигляді рядка.

void saveOrder(Order order) – Зберігає або оновлює замовлення.

Optional<Order> findById(Long id) – Знаходить замовлення за числовим ідентифікатором.

List<Order> getOrdersByReceiverId(Long receiverId) – Повертає список замовлень для конкретного отримувача.

List<Order> getOrdersByReceiverIdAndStatus(Long receiverId, String status) – Повертає список замовлень для отримувача з конкретним статусом.

List<OrderStatusHistory> getStatusHistoryByOrderId(Long orderId) – Отримує історію статусів конкретного замовлення.

Long countOrdersByUserId(Long userId) – Підраховує кількість замовлень, пов’язаних із користувачем.

List<Order> getOrdersByUserId(Long userId) – Повертає список замовлень, створених користувачем.

List<OrderStatusHistory> getStatusHistoryByUserId(Long userId) – Повертає історію статусів замовлень користувача.

boolean updateOrderStatus(Long orderId, String newStatus) – Оновлює статус замовлення.

List<OrderStatusHistory> getLastStatusByUserId(Long userId) – Отримує останні статуси по замовленнях користувача.

List<OrderLastStatusDateDto> getLastStatusDateByUserId(Long userId) – Отримує інформацію про останні дати зміни статусів замовлень користувача.

List<OrderStatusHistory> getLatestStatusHistoryByUserId(Long userId) – Повертає найактуальнішу історію статусів замовлень користувача.

Специфікація класу «OrderServiceImpl»

Ім’я: OrderServiceImpl.

Відповідальність: реалізує бізнес-логіку для створення, оновлення, перегляду та обробки замовлень. Забезпечує управління статусами замовлень, роботу з чернетками, а також історію змін.

Атрибути:

OrderRepository – доступ до бази даних замовлень.

UserRepository – доступ до інформації про користувачів (відправників/отримувачів).

AddressesRepository – доступ до збережених адрес доставлення.

TechnicCharacterCargoService – доступ до сервісу для отримання Інформації про вантажі.

StatusHistoryRepository – доступ до історії змін статусів замовлень.

Операції зі специфікаціями нетривіальних операцій:

saveDraftOrder(OrderForm form) – Створює чернетку замовлення з початковим статусом DRAFT, зберігає її в базу, повертає ID замовлення.

finalizeOrder(Long orderId) – Переводить чернетку в статус TO_FEED, якщо всі узгодження завершено.

updateOrderStatus(Long orderId, String newStatus) – Оновлює статус замовлення, зберігаючи історію змін.

getOrderById(String id) – Повертає повну інформацію про замовлення за його ID з ініціалізацією залежностей.

getOrdersByUserId(Long userId) – Повертає список замовлень, де користувач є відправником або отримувачем.

getStatusHistoryByUserId(Long userId) – Повертає всі записи історії статусів для користувача.

getLastStatusDateByUserId(Long userId) – Повертає дату останньої зміни статусу кожного замовлення користувача.

Для кращого розуміння динаміки взаємодії між об’єктами в системі використовуються діаграми послідовності. Вони дозволяють наочно відобразити послідовність обміну повідомленнями, що відбувається під час виконання певного процесу або сценарію користування. Такий підхід дає змогу проаналізувати логіку роботи програмного забезпечення в часовому аспекті та виявити можливі недоліки у взаємодії її компонентів.

Основними елементами діаграми послідовності є позначення об'єктів (прямокутники з назвами об'єктів), вертикальні «лінії життя» (англ. Lifeline), що відображають плин часу, прямокутники, що відображають діяльність об'єкта або виконання ним певної функції (прямокутники на пунктирну «лінії життя»), і стрілки, що показують обмін сигналами або повідомленнями між об'єктами [10].

На рисунку 2.7 представлено діаграму послідовності «Створення та збереження історії замовлення на вантажоперевезення».

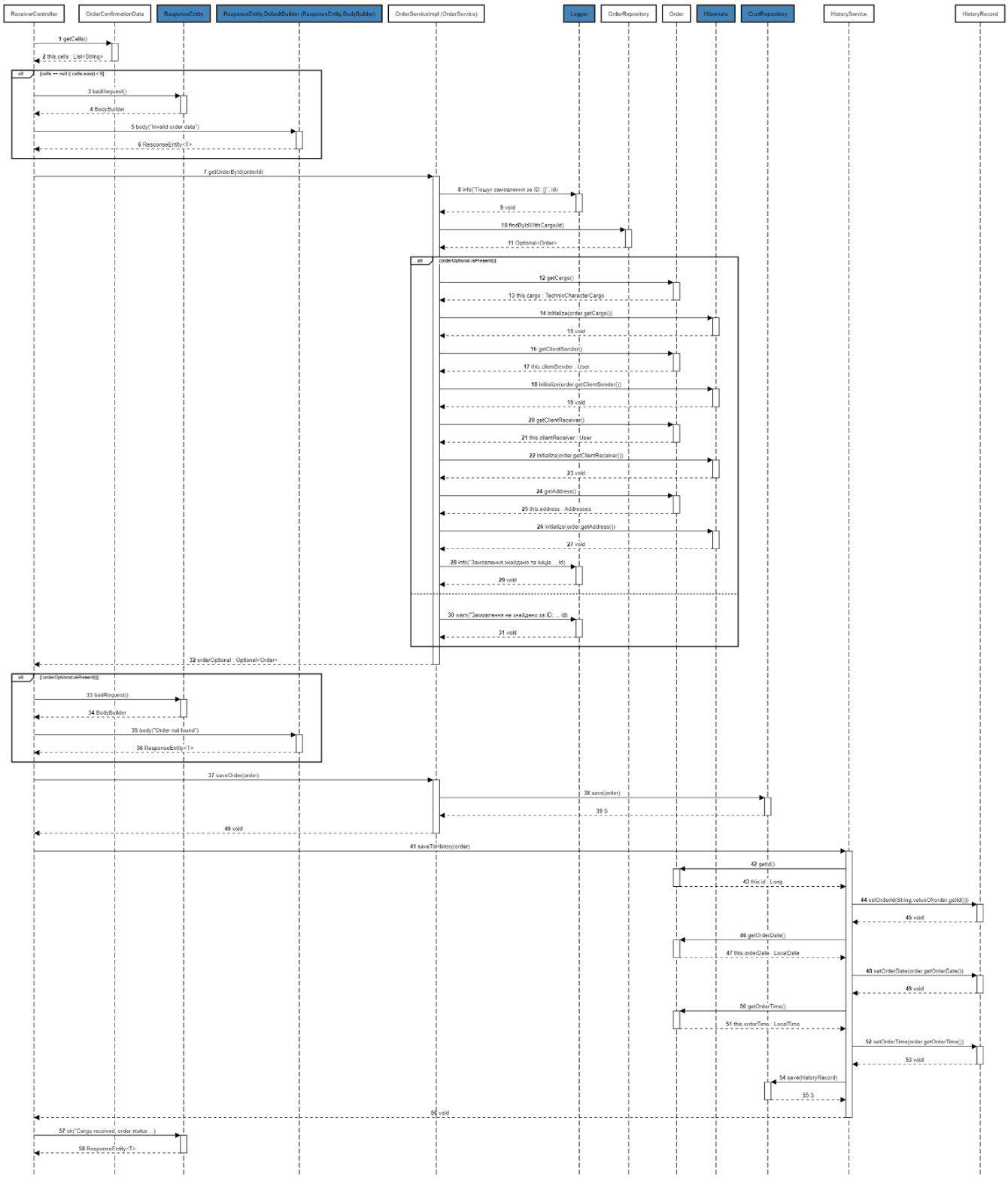


Рисунок 2.7 – Діаграма послідовності «Створення та збереження історії замовлення на вантажоперевезення»

Ця діаграма послідовності деталізує процес створення та збереження історії змін після оформлення замовлення на вантажоперевезення. Вона показує, як після

створення об’єкта замовлення виконується збереження історії, зокрема – хто, коли і які дії виконав. Починається все з того, що користувач надсилає заявку через контролер RouteController. Далі, через сервіси валідації й створення, формується об'єкт Order, який зберігається в базу даних за допомогою OrderRepository. Цей процес описаний у верхній частині діаграми. Після того, як замовлення створено і збережено, ініціюється логування змін.

Вебзастосунок створює запис історії через HistoryService, який формує об'єкт HistoryRecord. Тут важливим є те, що інформація, яка записується в історію, включає дані про замовлення, його статус, користувача, який його створив, дату та тип події. Щоб сформувати запис історії, HistoryService отримує поточного користувача через Logger, а також отримує ID замовлення. Потім через сервіси обробляє, що саме змінилося, і створює відповідний запис історії в базі даних через HistoryRepository.

На діаграмі видно, що після збереження замовлення викликається метод saveHistoryRecord(), який працює з новоствореним об’єктом типу HistoryRecord. Потім цей запис також зберігається в базу через відповідний репозиторій. Таким чином, діаграма демонструє, що збереження історії змін це окремий логічний етап, який відбувається незалежно від збереження самого замовлення, але одразу після нього. Такий підхід дозволяє відстежувати всі зміни та дії в програмному застосунку для подальшого аудиту, контролю або просто перегляду.

У результаті ця діаграма показує не просто створення замовлення, а й важливий аспект супроводу – формування історії змін, що є обов’язковим для будь-якої сучасної облікової або логістичної системи. Вона підкреслює важливість розмежування обов’язків у коді: контролер відповідає за приймання запиту, сервіси – за бізнес-логіку, а репозиторії – за взаємодію з базою.

На рисунку 2.8 представлено діаграму послідовності «Узгодження замовлення на вантажоперевезення»

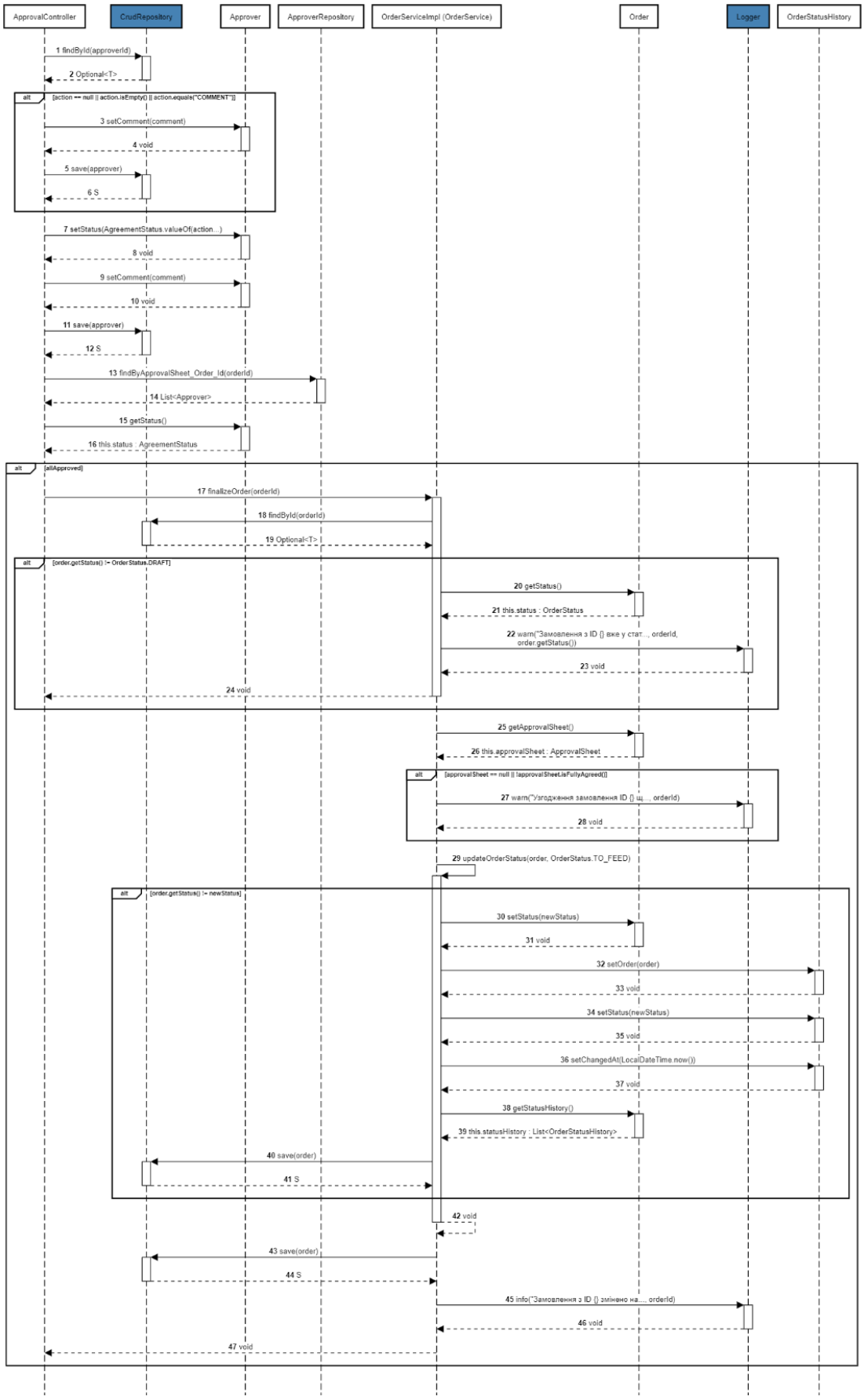


Рисунок 2.8 – Діаграмма послідовності «Узгодження замовлення на вантажоперевезення»

Ця діаграма послідовності відображає процес узгодження замовлення, починаючи з моменту, коли користувач надсилає коментар до одного з етапів погодження, і закінчуючи оновленням статусу замовлення в базі даних та збереженням історії змін. У ній залучені такі основні компоненти: контролер ApprovalController, репозиторії CrudRepository і ApproveRepository, сервіс OrderServiceImpl, об’єкт Order, клас для логування Logger та сутність OrderStatusHistory, яка відповідає за збереження історії змін статусів замовлень.

Процес починається з того, що контролер отримує погодження за ідентифікатором. Якщо обрана секція має значення "COMMENT", то створюється об’єкт коментаря, зберігається в базі, після чого оновлюється статус погодження, і цей статус також зберігається. Далі вебзастосунок перевіряє всі погодження, пов’язані із цим замовленням, і визначає їх загальний статус. Якщо всі погодження були затверджені, викликається метод finalizeOrder, який починає процес зміни статусу самого замовлення.

Сервіс знаходить відповідне замовлення за ID і перевіряє його поточний статус. Якщо воно ще перебуває в статусі "чернетка" (DRAFT), то вебзастосунок намагається отримати пов'язаний з ним лист погодження. Якщо він існує, то статус замовлення оновлюється на TO_FEED. Після цього в об’єкті Order оновлюється поле зі статусом, а також фіксується дата зміни. Вебзастосунок витягує історію змін статусів і додає новий запис про зміну в OrderStatusHistory. Потім змінений об’єкт замовлення зберігається, і у логах фіксується повідомлення про успішну зміну статусу.

Важливо, що на кожному етапі діаграма враховує можливість помилок або нестандартних ситуацій: наприклад, якщо замовлення вже не в статусі DRAFT, або якщо не вдалося знайти лист погодження, то вебзастосунок виводить попередження у лог, але не виконує зміну статусу. Таким чином забезпечується стабільність роботи бізнес-логіки. На рисунку 2.9 представлено діаграму послідовності «Отримання статистики замовлень користувача»

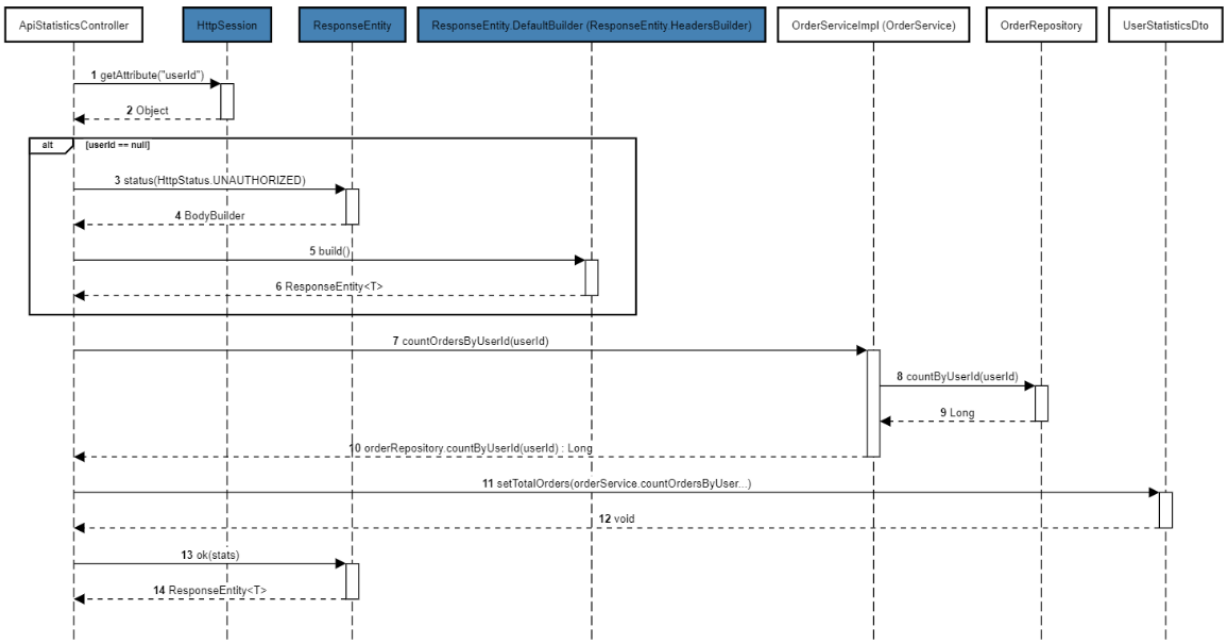


Рисунок 2.9 – Діаграма послідовності «Отримання статистики замовлень користувача»

Ця діаграма послідовності ілюструє логіку обробки запиту для отримання статистики замовлень користувача у контролері ApiStatisticsController. Вона демонструє, як вебзастосунок перевіряє наявність автентифікованого користувача через сесію, звертається до сервісу для підрахунку кількості замовлень, і повертає відповідь у вигляді HTTP-відповіді (ResponseEntity).

На початку ApiStatisticsController звертається до об’єкта HttpSession, щоб отримати значення атрибута "userId" (поз. 1). Якщо userId дорівнює null, виконується альтернативний сценарій, у якому формується відповідь зі статусом 401 Unauthorized (поз. 3–6). Це забезпечує базову перевірку доступу: якщо користувач не авторизований, він не отримає статистику.

У випадку, якщо userId не null, вебзастосунок передає його до методу countOrdersByUserId у сервісі OrderServiceImpl (поз. 7), який далі делегує запит до OrderRepository (поз. 8). Репозиторій виконує запит до бази даних, щоб підрахувати кількість замовлень, пов’язаних з цим користувачем (поз. 9), і повертає значення типу Long.

Після отримання кількості замовлень сервіс встановлює це значення у DTO об’єкт UserStatisticsDto методом setTotalOrders (поз. 11–12). Далі контролер формує успішну відповідь зі статусом 200 OK, вкладаючи туди зібрану статистику (поз. 13–14).

2.4 Реалізація вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion”.

Контролер SendCargoController, який реалізує функціональність створення замовлення на відправдення вантажу в вебзастосунку, побудованому на Spring Framework. Контролер обробляє як GET-запит, що відкриває сторінку створення замовлення, так і POST-запит, який відповідає за збереження введених даних. У методі getSendCargoPage створюються об’єкти форм CargoForm та OrderForm, які потім передаються у шаблон для заповнення користувачем. Також із бази даних витягується список користувачів, при цьому виключається поточний користувач, щоб він не міг сам собі відправити вантаж. Метод sendCargo приймає дані, що надійшли з форми, і перевіряє наявність користувача-ініціатора через його email. Якщо користувача знайдено, визначаються ID відправника та отримувача. Далі, за допомогою відповідних сервісів, зберігаються адреса доставлення та технічні характеристики вантажу.

```
    @GetMapping("/send-cargo")
    public String getSendCargoPage(Model model, Principal principal) {
        CargoForm cargoForm = new CargoForm();
        OrderForm orderForm = new OrderForm();

        model.addAttribute("cargoForm", cargoForm);
        model.addAttribute("orderForm", orderForm);
        Optional<User> currentUserOpt =
userRepository.findByEmail(principal.getName());

        List<User> users;
        if (currentUserOpt.isPresent()) {
            User currentUser = currentUserOpt.get();
            users = userRepository.findAll()
                    .stream()
                    .filter(user -> !user.getId().equals(currentUser.getId()))
                    .collect(Collectors.toList());
        } else {
            users = userRepository.findAll();
```

```
        }
        model.addAttribute("users", users);
        return "send-cargo";
    }
```

Потім ці дані прикріплюються до форми замовлення, яка передається у метод збереження чернетки замовлення. У разі успішного створення замовлення створюється аркуш погодження з двома сторонами: відправником і отримувачем. Крім того, фіксується дата замовлення у форматі dd.MM.yyyy. У випадку помилки в процесі збереження, користувач перенаправляється назад на сторінку створення замовлення із повідомленням про помилку. У контролері активно використовуються сервіси для роботи з базою даних, а також система логування для виведення інформації про хід виконання або проблеми.

Загалом, SendCargoController відповідає за логіку створення нового логістичного замовлення із урахуванням користувацьких даних, адрес доставлення, характеристик вантажу та наступного погодження між сторонами.

Клас ReceiverController є Spring MVC контролером, який обробляє запити, пов’язані з отриманням вантажу користувачем у ролі отримувача. Він надає функціонал для перегляду замовлень, підтвердження отримання вантажу, а також перегляду історії замовлень і статусів. Для отримання списку замовлень за маршрутом GET /service/receive-cargo контролер визначає авторизованого користувача через об’єкт Principal, звертається до сервісу OrderService, отримує замовлення, де користувач є отримувачем, і передає їх у модель для відображення на сторінці. POST-запит на той самий маршрут приймає дані підтвердження замовлення у вигляді JSON, перевіряє їх, знаходить відповідне замовлення за ID, зберігає оновлене замовлення і записує зміни в історію через HistoryService, після чого повертає відповідь із статусом операції.

На рисунку 2.10 представлено головну сторінку.

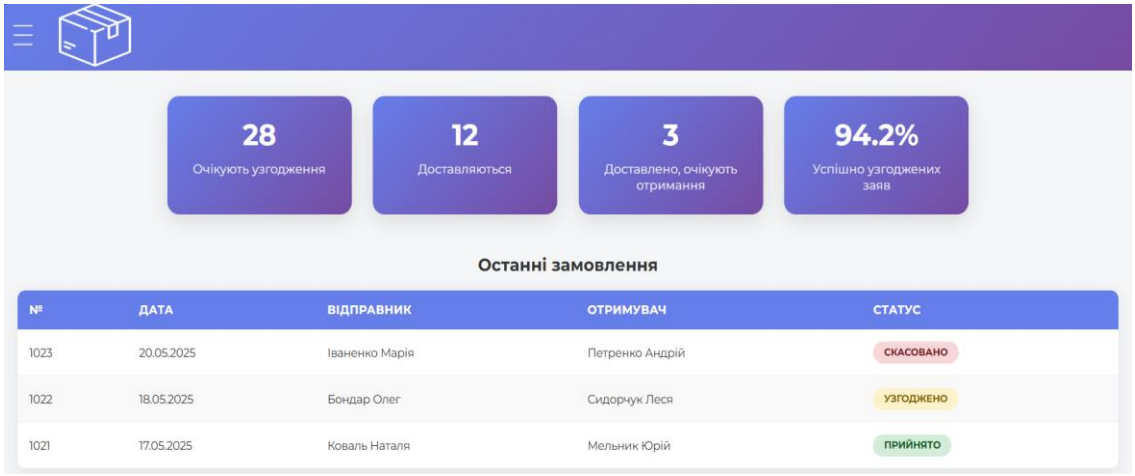


Рисунок 2.10 – Головна сторінка вебзастосунка

На зображенні представлено фрагмент інтерфейсу вебзастосунка вантажоперевезень. У верхній частині відображена зведена статистика: кількість заявок на узгодженні (28), тих, що доставляються (12), та тих, що очікують отримання (3), а також відсоток успішно узгоджених заяв 94.2%. Нижче наведена таблиця з останніми замовленнями, де вказано номер, дату, відправника, отримувача та статус кожного замовлення (наприклад, "Скасовано", "Узгоджено", "Прийнято"). Інтерфейс виконано у сучасному стилі з використанням кольорових позначень для зручності сприйняття.

2.5 Тестування вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion”.

Забезпечення якості програмного забезпечення є одним із ключових завдань у процесі розробки. Щоб зрозуміти, що саме мається на увазі під терміном «якість програмного продукту», слід враховувати низку важливих аспектів.

Якість – це ступінь відповідності системи, компоненту або процесу заданим вимогам, потребам або очікуванням користувача. Щоб визначити надійність, компоненту або процесу використовують так звані атрибути якості,

характеристики, що відображають певну властивість; найпоширеніший спосіб контролю якості будь-якого програмного продукту – належне тестування [11];

Мета тестування - виявлення помилок чи неузгодженостей. Інакше кажучи, потрібно знайти помилки (локалізація – задача діагностики) та зробити все для їхнього усунення [11].

Основні завдання, які ставилися під час тестування: перевірити, чи відповідає функціонал програми очікуваному результату згідно з технічним завданням; виявити й усунути критичні помилки, які можуть вплинути на стабільність роботи; переконатися, що основні модулі системи працюють правильно та взаємодіють між собою без збоїв; забезпечити зручність використання системи для кінцевого користувача.

Таким чином, тестування допомагає покращити якість програмного продукту та забезпечити його надійну роботу в майбутньому.

Основні підходи з тестування програмного забезпечення:

Black-box testing – тестування без знання внутрішньої реалізації, орієнтоване на перевірку функціональності за вхідними та вихідними даними.

White-box testing – тестування з доступом до коду, спрямоване на перевірку логіки, умов, гілок та циклів.

Unit testing – тестування окремих модулів або функцій, зазвичай автоматизоване; використовується в white- або grey-box підходах.

System testing – перевірка всієї системи відповідно до вимог, охоплює функціональні й нефункціональні тести.

Performance testing – оцінка швидкодії, навантаження та стабільності системи.

Security testing – виявлення вразливостей, перевірка механізмів захисту, аутентифікації та авторизації.

В таблиці 2.1 представлено тест-кейси чорної скриньки (повну таблицю представлено в додатку Д).

Таблиця 2.1 – Тест-кейси методу чорної скриньки

ID тесту	Назва	Мета	Кроки	Очікуваний результат	Фактичний результат
TC-BB-001	Відправлення заявки без вибору отримувача	Перевірити валідацію, якщо користувач не вибрав отримувача	1. Увійти в систему 2. Перейти на сторінку «Відправити вантаж» 3. Заповнити форму, але не вказати отримувача 4. Натиснути “Надіслати заявку”	Виводиться повідомлення: Оберіть один із пунктів списку	Виводиться повідомлення: Оберіть один із пунктів списку
TC-BB-002	Реєстрація з email без "@"	Перевірити, що не можна зареєструватися з некоректною email-адресою	1. Перейти на сторінку реєстрації 2. Заповнити всі поля, але email = testemail.com3. Натиснути “Зареєструвати”	З’являється повідомлення про помилку формату email. Реєстрація не відбувається.	З’являється повідомлення про помилку формату email. Реєстрація не відбувається.

В таблиці 2.2 представлено тест-кейси білої скриньки реалізовані за допомогою JUnit 5 (повну таблицю представлено в додатку Д).

Таблиця 2.2 – Тест-кейси методу білої скриньки

ID тесту	Назва тесту	Вхідні дані	Початкові умови	Очікуваний результат	Покриття гілки	Фактичний результат
TC-WB-001	Зміна статусу на новий, валідний	orderId = 1, newStatusString = "TO_FEED"	Order зі статусом DRAFT	Повертає true, статус оновлено, історія збережена	order.getStatus() != newStatus → true	Статус замовлення з ID 1 оновлено на TO_FEED
TC-WB-002	Замовлення не знайдено	orderId = 9999, newStatusString = "DONE"	Замовлення з таким ID не існує	Повертає false	orderOpt.isEmpty() → true	Замовлення з ID 9999 не знайдено

В таблиці 2.3 представлено тест-кейси методу системного тестування (повну таблицю представлено в додатку Д).

Таблиця 2.3 – Тест-кейси методу системного тестування

ID тесту	**Назва тесту**	**Мета**	**Кроки**	**Очікуваний результат**	**Перевіряються компоненти**	**Фактичний результат**
TC-ST-001	Повний цикл: створення заяви на вантажоперевезення, її узгодження відповідальними сторонами	Перевірити узгоджену роботу всіх модулів системи при створенні заявки	1. Увійти 2. Перейти на сторінку «Відправити вантаж» 3. Заповнити форму та завершити оформлення 4. Узгодити заяву (як відправник) 5. Узгодити заяву (як отримувач)	Замовлення створено та погоджено, статус установлено TO_FEED, зміна статусу записано в історію	UI, контролери, сервіси, база, авторизація, логіка	Замовлення створено та погоджено, статус установлено TO_FEED, зміна статусу записано в історію
TC-ST-002	Отримання вантажу й підтвердження доставлення	Перевірити правильну зміну статусу на DONE та фіксацію в історії	1. Увійти як отримувач 2. Перейти на «Отримати вантаж» 3.Вибрати заявку 4.Натиснути “Підтвердити отримання”	Статус змінено на DONE, замовлення має зникнути з таблиці вантажів для отримання та з’явитись в таблиці «Історія замовлення»	Контролери, БД, логіка, сервіс історії	Статус змінено на DONE, замовлення має зникнути з таблиці вантажів для отримання та з’явитись в таблиці «Історія замовлення»

В таблиці 2.4 представлено тест-кейси методу тестування продуктивності (повну таблицю представлено в додатку Д).

Таблиця 2.4 – Тест-кейси методу тестування продуктивності

ID тесту	**Назва тесту**	**Мета**	**Методика**	**Очікуваний результат**
TC-PT-001	Тест навантаження: 100 користувачів отримують список заявок	Перевірити стабільність при одночасних запитах до /api/user/{id}/orders	Імітація 100 паралельних запитів GET через JMeter	Середній час відповіді < 2 секунд, без помилок HTTP 5xx
TC-PT-002	Тест швидкодії: створення заявки через форму	Виміряти час, за який обробляється POST /send-cargo	Відправити 50 POST-запитів із валідними формами, виміряти середній час відповіді	Сервер обробляє форму < 1.5 секунд, жоден запит не падає

В таблиці 2.5 представлено тест-кейси методу тестування безпеки (повну таблицю представлено в додатку Д).

Таблиця 2.5 – Тест-кейси методу тестування безпеки

ID тесту	**Назва тесту**	**Мета**	**Кроки / Методика**	**Очікуваний результат**
TC-SEC-001	Спроба доступу до /api/user/{id} без авторизації	Перевірити, що API захищене і вимагає логін	Відправити GET-запит без JWT або сесії	Отримано HTTP 401 Unauthorized
TC-SEC-002	Користувач без прав намагається отримати чужу заявку	Перевірити, що користувач не бачить чужих даних	Авторизуватись як User A, зробити запит на /api/user/{id=іншого}	Отримано HTTP 403 Forbidden або 404 Not Found

Тестування вебзастосунка проведено комплексно із застосуванням методів чорної та білої скриньки, системного, продуктивного та безпекового тестування. Перевірено коректність функціоналу, логіку зміни статусів, узгодженість модулів, стійкість до навантаження та захист API. Усі тести пройдено успішно, що підтверджує відповідність системи вимогам, стабільну роботу та готовність до експлуатації.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ЛОГІСТИЧНИХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ “AGROFUSION”

3.1 Загальна характеристика реалізованого ПЗ

У процесі виконання кваліфікаційної роботи було здійснено повний цикл розробки вебзастосунка для автоматизації внутрішніх логістичних перевезень компанії “Agrofusion”. Робота включала аналіз вимог, проєктування архітектури, безпосередню реалізацію та тестування. У цьому розділі наводиться підсумковий опис отриманих результатів, ступінь реалізації функцій, кількісні та якісні характеристики ПЗ, технічні вимоги, а також результати випробувань згідно з методикою, наведеною в додатку Б.

Розроблене програмне забезпечення являє собою повнофункціональний вебзастосунок, призначений для автоматизації обробки внутрішніх перевезень вантажів між підрозділами компанії. У ході розробки враховано специфіку аграрного бізнесу, сезонні навантаження, необхідність оперативної обробки заявок, формування документації, ведення звітності, а також контроль за виконанням перевезень. Застосунок побудований за архітектурним патерном MVC з розділенням на три основні компоненти:

Model – сутності та бізнес-логіка (наприклад, об’єкти заявок, користувачів, статусів, ТТН);

View – користувацький інтерфейс, створений з використанням HTML, CSS та шаблонізатора Thymeleaf;

Controller – обробка запитів, взаємодія з сервісами, маршрутизація.

Для реалізації проєкту використано стек технологій: Java 17, Spring Boot, Spring Security, Thymeleaf, MySQL, Hibernate/JPA.

Застосунок передбачає наявність ролі користувача – логіст. Передбачено авторизацію, захист паролів, ведення історії змін статусів.

3.2 Технічні та кількісні характеристики

Загальна кількість вихідного коду у проєкті становить 76 569 рядків, з яких 9 100 є активними (без коментарів і порожніх рядків). Найбільше коду написано мовами HTML (4880 рядків) та Java (2527 рядків), що відповідає веборієнтованому характеру застосунку. Файлова структура згідно з стандартом Spring Boot.

Загальний обсяг вихідного коду (у рядках) представлено на рисунку 3.1.

file type	LOC	NCLOC
HTML	4 880	4 783
Java	2 527	2 299
Cascading style sheet	834	833
JavaScript	502	491
Shell script	278	278
XML	187	187
Files supported via TextMate bun	169	169
.gitignore (GitIgnore)	29	29
Properties	16	16
YAML	9	9
Markdown	6	6
Text	67 132	
Total	**76 569**	**9 100**
Average	6 380,75	827,27

Рисунок 3.1 – Загальний обсяг вихідного коду

3.3 Результати тестування

Тестування проводилось згідно з методикою (додаток Д) і включало: функціональне тестування: перевірено основні сценарії. Збоїв не виявлено. Навантажувальне тестування: при 100 одночасних запитах вебзастосунок відповідав з часом затримки до 350 мс. В таблиці 3.1 представлено середні часові характеристики операцій.

Таблиця 3.1 – Середні часові характеристики операцій

Операція	**Час виконання**
Авторизація	~0.8 с
Створення заявки	~2.5 с
Генерація ТТН	~1.8 с
Формування звіту	~2.0 с

Через обмеження обсягу попередніх розділів не були повністю описані наступні реалізовані можливості:

– Система логування дій користувачів (реалізована через Spring Boot Logger);

– REST API для отримання інформації про користувачів, адреси, вантажі (див. ApiController в додатку Б);

– Механізм фіксації змін статусів замовлень із точним часом зміни (таблиця order_status_history).

Таким чином, у межах кваліфікаційної роботи було повністю реалізовано та протестовано вебзастосунок, що відповідає потребам компанії “Agrofusion” у сфері автоматизації внутрішніх перевезень. Програмне забезпечення готове до подальшого розгортання, підтримки та вдосконалення.

4 ОХОРОНА ПРАЦІ

4.1 Загальні положення з охорони праці

Охорона праці − це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [12].

Роботодавець − власник підприємства, установи, організації або уповноважений ним орган, незалежно від форм власності, виду діяльності, господарювання, і фізична особа, яка використовує найману працю.

Працівник − особа, яка працює на підприємстві, в організації, установі та виконує обов'язки або функції згідно з трудовим договором (контрактом). Державна політика в галузі охорони праці визначається відповідно до Конституції України Верховною Радою України і спрямована на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням [12].

Державна політика в галузі охорони праці базується на принципах: − пріоритету життя і здоров'я працівників, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці; − підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;

− комплексного розв'язання завдань охорони праці на основі загальнодержавної, галузевих, регіональних програм з цього питання та з урахуванням інших напрямів економічної і соціальної політики, досягнень в галузі науки і техніки та охорони довкілля;

− соціального захисту працівників, повного відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;

− встановлення єдиних вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності;

− адаптації трудових процесів до можливостей працівника з урахуванням його здоров'я та психологічного стану;

− використання економічних методів управління охороною праці, участі держави у фінансуванні заходів щодо охорони праці, залучення добровільних внесків та інших надходжень на ці цілі, отримання яких не суперечить законодавству;

− інформування населення, проведення навчання, професійної підготовки і підвищення кваліфікації працівників з питань охорони праці;

− забезпечення координації діяльності органів державної влади, установ, організацій, об'єднань громадян, що розв'язують проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництва і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці на місцевому та державному рівнях;

− використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародного співробітництва.

4.2 Структура управління питаннями охорони праці на підприємстві

В Україні, на сьогодні, функціонує багаторівнева система управління охороною праці, функціональними ланками якої є відповідні структури державної законодавчої та виконавчої влади різних рівнів, управлінські структури підприємств, трудові колективи, професійні спілки, добровільні громадські об'єднання працівників і спеціалістів з охорони праці. За спрямованістю комплексу розв'язаних питань усі ланки системи управління охороною праці можна розділити на дві групи: Перша група - ланки, які забезпечують вирішення законодавчо-

нормативних, науково-технічних, соціально-економічних та інших загальних питань охорони праці. До першої групи ланок системи управління охороною праці входять органи державної законодавчої ініціативи й органи державного управління в галузі охорони праці. Друга група – ланки, у функціональні обов'язки яких входить забезпечення безпеки праці в умовах конкретних галузей і виробництв.

До другої групи ланок системи управління охороною праці входять управлінські структури підприємств які забезпечують в умовах конкретних виробництв реалізацію вимог законодавчих і нормативних актів про охорону праці з метою створення безпечних і нешкідливих умов праці, попередження виробничого травматизму та професійних захворювань, вирішують весь комплекс питань з охорони праці, пов'язаних з даним виробництвом.

У своїй діяльності щодо охорони праці управлінські структури підприємств взаємодіють з комісією з питань охорони праці підприємства (при наявності такої), з профспілками підприємства та уповноваженими трудових колективів. Чисельні наглядові й контрольні органи не можуть гарантувати повну безпеку ведення робіт на виробництві. Для розв’язання проблем у сфері охорони праці на підприємстві повинна діяти ефективна система управління охороною праці. 12 грудня 2018 року розпорядженням Кабінету Міністрів України № 989-р схвалена Концепція реформування системи управління охороною праці в Україні. Ця Концепція визначає принципи, основні напрями та завдання побудови системи організації безпеки та гігієни праці в Україні на основі ризикоорієнтованого підходу для забезпечення впровадження стандартів Європейського Союзу. Метою Концепції є створення національної системи запобігання виробничим ризикам для забезпечення ефективної реалізації права працівників на безпечні та здорові умови праці. Згідно з Рекомендаціями, система управління охороною праці на підприємстві – це ряд комплексних заходів та механізмів, які повинні забезпечувати безпечність трудової діяльності робітників у сфері технологічного, санітарного, профілактичного та організаційного стану виробничого середовища компанії [13].

В таблиці 4.1 перераховані шкідливі та небезпечні виробничі фактори, з якими можуть зіткнутися під час роботи працівники.

Таблиця 4.1 – Шкідливі та небезпечні виробничі фактори на підприємстві

Хімічна	**Фізичні**	**Біологічні**	**Психологічні**
Токсичні речовини, пил, пара, газ	Параметри повітря у приміщенні (температура, вологість, швидкість руху повітря), вібрація, шум, нетоксичний пил, пара, різні види випромінювань, освітленість	Мікроорганізми, бактерії, інфекції	Фізичні та нервово-психологічні перевантаження, монотонність праці, емоційне перевантаження.

Шкідливі й небезпечні виробничі фактори – це ті фактори, які в результаті свого тривалого або короткочасного впливу на людину призводять до погіршення стану його здоров’я або до травми[14].

Небезпечними називаються чинники, здатні при відповідних умовах викликати гостре порушення здоров’я або загибель організму; шкідливими – чинники, що чинять негативний вплив на працездатність або викликають професійні захворювання і інші професійні наслідки.

На рисунку 4.1 представлені наслідки різних виробничих факторів.

Рисунок 4.1 – Наслідки різних видів виробничих факторів

Шкідливий виробничий фактор діє переважно тривалий час або має накопичувальний ефект. Його наслідками є розвиток професійних або хронічних

захворювань, поступове зниження працездатності, а також потенційний негативний вплив на репродуктивне здоров’я, що може відобразитися на здоров’ї майбутнього покоління.

Небезпечний виробничий фактор – це такий, що при дії на працівника може миттєво призвести до серйозних наслідків: фізичних травм різного ступеня тяжкості (порізи, переломи, опіки тощо), гострих отруєнь або навіть до летального випадку. Такі наслідки зазвичай виникають раптово та потребують негайного реагування.

ВИСНОВКИ

Мета кваліфікаційної роботи розробка вебзастосунка для внутрішніх перевезень компанії “Agrofusion” була успішно досягнута. У ході виконання кваліфікаційної роботи було виконано наступні завдання:

1) проаналізовано задачу інженерії програмного забезпечення з розробки вебзастосунка для внутрішніх перевезень компанії "Agrofusion";

2) проаналізовано наявне програмне забезпечення для управління логістикою компаній;

3) поставлено задачу на розробку вебзастосунка для внутрішніх перевезень компанії "Agrofusion";

4) розроблено архітектуру вебзастосунка для внутрішніх перевезень компанії “Agrofusion”;

5) розроблено проєкт вебзастосунка для внутрішніх перевезень компанії “Agrofusion”;

6) програмно реалізовано вебзастосунок для внутрішніх перевезень компанії “Agrofusion”;

7) виконано тестування вебзастосунка для внутрішніх перевезень компанії “Agrofusion”.

Отримані результати стали завершальним етапом кваліфікаційної роботи, в межах якого було успішно реалізовано вебзастосунок відповідно до поставлених вимог. Розроблене технічне завдання дало змогу чітко організувати процес проєктування та впровадження програмного забезпечення з урахуванням потреб бізнесу й логістичних підрозділів компанії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Портал компанії "Agrofusion". URL: https://www.inagro.ua/uk/main/ (дата звернення: 24.02.2025).

2. Пилипенко І. О. Специфіка підходів до реалізації проєктів з різним ступенем невизначеності. URL: https://nau.edu.ua/site/variables/docs/docsmenu/studnauka/general.pdf#page=23 (дата звернення: 26.02.2025).

3. Oracle Transportation Management (OTM). URL: https://www.oracle.com/scm/logistics/transportation-management/ (дата звернення: 03.03.2025).

4. Blue Yonder. URL: https://es.blueyonder.com/about (дата звернення: 07.03.2025).

5. Свінцицька О. М., Граф М. С., Нікітчук Т. М. Метод Use Case в плануванні проєктів з інформаційних технологій. Технічна інженерія. 2022. № 89. С. 77-84.

6. Штогрин Д. А. Роль патернів проєктування в розробці архітектури ПЗ. Концептуальні проблеми розвитку сучасної гуманітарної та прикладної науки: матеріали VI Міжнародного науково-практичного симпозіуму. 2022. С. 395-398. URL: http://repository.ukd.edu.ua/bitstream/handle/123456789/306/Штогрин%20Д%20симпозіум%202022.pdf?sequence=1&isAllowed=y (дата звернення: 12.05.2025).

7. Asieieva A. Аналіз проблем вибору технології для розробки програмного забезпечення. Computer-integrated technologies: education, science, production. 2019. № 37. С. 10-18.

8. Костюк Ю. В. Використання діаграми діяльності для моделювання та аналізу процесу прийняття рішень. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення: матеріали Міжнародної наукової

інтернет-конференції (м. Тернопіль, Україна – м. Переворськ, Польща, 8-9 червня 2023 р.). Тернопіль: ГО "Наукова спільнота", 2023. Вип. 78. С. 60.

9. Шаров С. В., Шамардак О. А. Концептуальне моделювання інформаційної системи за допомогою мови UML. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. 2014. № 7. С. 10-14.

10. Єфремов М. Ф., Єфремов Ю. М., Єфремов В. М. Проєктування програмного забезпечення з використанням UML. Житомир: ЖДТУ, 2016. URL: https://eztuir.ztu.edu.ua/bitstream/handle/123456789/3296/12.pdf?sequence=1 (дата звернення: 08.05.2025).

11. Рудик І., Шевчук Б. Потреба тестування програмного забезпечення. Матеріали конференцій МНЛ (м. Біла Церква, 7 жовтня 2022 р.). Біла Церква, 2022. С. 186-188.

12. Про охорону праці: Закон України від 14.10.1992 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 13.05.2025).

13. Курепін В. М. Особливості системи управління охороною праці в аграрних підприємствах: економічні аспекти розвитку. Modern Economics. 2021. № 29. С. 107-114. DOI: https://doi.org/10.31521/modecon.V29(2021)-17.

14. Вавженчук С. Правовий статус працівника як суб'єкта охоронних трудових правовідносин. Підприємництво, господарство і право. 2016. № 7. С. 55-59.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ «AGROFUSION»

1 Вступ

1.1 Назва програмного забезпечення

Вебзастосунок для внутрішніх перевезень компанії “Agrofusion”.

1.2 Сфера застосування

Логістика.

2 Підстава для розробки програмного забезпечення

Розробка програмного забезпечення виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

3 Призначення розробки

3.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація операції зі створення заявки на перевезення вантажів між внутрішніми підрозділами компанії, генерація документів до цих заяв, облік та звітність.

3.2 Експлуатаційне призначення

Експлуатація вебзастосунка буде здійснюватися логістом внутрішніх перевезень. Основні задачі які будуть виконуватися в застосунку це створення та контроль заявок на вантажоперевезення, введення та облік відповідних документів та звітність. Призначений для щоденного користування в умовах офісної роботи. Доступ здійснюється через браузер з доступом до локальної мережі чи інтернету.

4 Технічні вимоги до програми

4.1 Вимоги до функціональних характеристик

4.1.1 Вимоги до переліку виконуваних функцій

Вебзастосунок повинен реалізовувати наступний перелік функцій:

- Створення заявок на перевезення, включаючи тип вантажу, обсяг, пункти відправлення та призначення, бажаний час доставлення.

- Погодження заявок відповідальною особою.
- Відстеження статусу заявок.
- Друк ТТН.
- Зберігання історії документів і можливість їх пошуку за різними критеріями.
- Можливість створення користувацьких звітів та експорту даних у різні формати.

4.1.2 Вимоги до організації вхідних та вихідних даних

Вебзастосунок повинен обробляти наступні вхідні дані, що вводяться користувачем через інтерфейс програмного забезпечення:

– Створення заявок на перевезення, включаючи тип вантажу, обсяг, пункти відправлення та призначення, бажаний час доставлення.

Вхідні дані: тип вантажу, обсяг вантажу, пункт відправлення, пункт призначення, спеціальні умови для перевезення (температурний режим, вентиляція, вологість, великогабаритний вантаж), замовник.

– Погодження заявок відповідальною особою.

Вхідні дані: заява на вантажоперевезення, погоджувальний лист.

– Відстеження статусу заявок.

Вхідні дані: узгоджена заява на вантажоперевезення.

– Формування стандартних звітів про виконані перевезення.

Вхідні дані: заява на вантажоперевезення.

– Можливість створення користувацьких звітів та експорту даних у різні формати.

Вхідні дані: заява на вантажоперевезення.

На основі введених вхідних даних вебзастосунок повинен генерувати наступні вихідні дані:

– Створення та редагування заявок на перевезення, включаючи тип вантажу, обсяг, пункти відправлення та призначення, бажаний час доставлення.

Вихідні дані: заява на вантажоперевезення.

– Погодження заявок відповідальною особою.

Вихідні дані: узгоджена заява на вантажоперевезення.

– Відстеження статусу заявок.

Вихідні дані: Етап вантажоперевезення узгодженої заяви.

– Можливість друку необхідних документів, наприклад, товарно-транспортних накладних.

Вихідні дані: Заява на вантажоперевезення.

– Зберігання історії документів і можливість їх пошуку за різними критеріями.

Вихідні дані: перелік всіх поданих документів, таких як заяви, ТТН, узгоджувальні та пояснювальні листи, із зазначенням дати подачі та статусу обробки, зокрема, чи документ узгоджено, чи не узгоджено, чи скасовано або є актуальним. Фільтрація документів за різними критеріями, такими як дата подачі або дата узгодження, статус документа (узгоджений, скасований, в обробці), тип документа (заява, ТТН, узгоджувальний лист, пояснювальний лист), а також перевізник або вантажоотримувач.

– Формування стандартних звітів про виконані перевезення.

Вихідні дані: Звіт про вантажоперевезення (з деталями про фіксовані статуси перевезення, час фіксації кожного етапу, час затримки).

– Можливість створення користувацьких звітів та експорту даних у різні формати.

Вихідні дані: Звіт про вантажоперевезення (залежно від потреб можна сформувати, наприклад за кількістю узгоджених або відхилених заяв.

4.2 Вимоги до надійності

Вебзастосунок повинен забезпечувати високий рівень надійності та стійкості до збоїв під час експлуатації. Вебзастосунок повинен функціонувати безперервно 24/7 з мінімальними перервами на технічне обслуговування, які не повинні перевищувати 2 години на місяць. Всі введені дані повинні зберігатися без втрат, з регулярними резервними копіями, які зберігаються не менше 30 днів. У разі збоїв

вебзастосунок повинен мати можливість відновлення даних до останньої успішної операції.

Вебзастосунок повинен коректно обробляти помилки та інформувати користувачів про неполадки через зрозумілі повідомлення, а всі помилки повинні логуватися для подальшого аналізу. У випадку аварії або збоїв вебзастосунок повинен автоматично переходити в резервний режим роботи, щоб забезпечити мінімальні втрати даних і доступ до критичних функцій. Вебзастосунок повинен бути масштабованим, щоб зберігати високу продуктивність при збільшенні навантаження, а моніторинг роботи системи має бути налаштований для виявлення проблем у реальному часі, з автоматичними сповіщеннями адміністраторів у разі критичних помилок. Для забезпечення безпеки даних вебзастосунок повинен мати захист від несанкціонованого доступу. Час відгуку програмного забезпечення для операцій, що не вимагають значної обробки даних, не повинен перевищувати 3 секунди, а для складних запитів – 5 секунд.

4.3 Умови експлуатації

4.3.1 Кліматичні умови експлуатації

Вебзастосунок повинен бути здатним функціонувати в умовах стандартних кліматичних умов для офісної роботи. Передбачено, що вебзастосунок буде використовуватися в приміщеннях з температурою навколишнього середовища від +10°C до +35°C, відносною вологістю повітря не більше ніж 80% без конденсації вологи. Вебзастосунок має бути здатним працювати на стандартних комп'ютерних системах, що використовуються в офісних умовах, без ризику збоїв, спричинених надмірною вологістю, низькими або високими температурами. У разі експлуатації за межами цих умов можуть виникати збої або зниження продуктивності системи.

4.3.2 Вимоги до чисельності та кваліфікації користувачів

Вебзастосунок буде використовуватися логістами внутрішніх перевезень компанії, а також адміністраторами системи. Чисельність користувачів повинна включати від 5 до 10 осіб у межах однієї організації залежно від розміру підприємства та кількості підрозділів, що здійснюють перевезення.

Для кожної категорії користувачів повинна бути передбачена відповідна навчальна програма, яка забезпечить ефективне використання функціоналу вебзастосунка. Враховуючи функціональні вимоги, вебзастосунок повинен бути інтуїтивно зрозумілим, що дозволить знизити потребу в спеціалізованому навчанні.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Вимоги до користувацького обладнання:

Для ефективної роботи вебзастосунка користувачі повинні мати відповідне обладнання, яке забезпечить належну функціональність системи. Основні вимоги до користувацького обладнання включають наступні елементи

Комп’ютери та ноутбуки. Вебзастосунок повинен бути сумісний з основними операційними системами, такими як Windows, macOS або Linux, із мінімальними вимогами до технічних характеристик: процесор не нижче ніж Intel Core i3 або еквівалент, 4 ГБ оперативної пам'яті (рекомендовано 8 ГБ), 2 ГБ вільного місця на жорсткому диску. Рекомендується використання сучасних браузерів, таких як Google Chrome, Mozilla Firefox або Microsoft Edge.

Інтернет-з’єднання. Для коректної роботи вебзастосунка необхідне стабільне підключення до Інтернету або до локальної мережі підприємства. Мінімальна швидкість інтернет-з’єднання повинна становити 1 Мбіт/с для безперебійної роботи системи, з оптимальною швидкістю 5 Мбіт/с або вище для великих обсягів даних і швидкої обробки запитів.

Браузери. Вебзастосунок оптимізований для роботи з останніми версіями браузерів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari. Використання застарілих версій браузерів може призвести до зниження швидкості роботи або некоректного відображення елементів інтерфейсу.

4.4.2 Вимоги до серверного обладнання

Для належної роботи вебзастосунка необхідно використовувати серверне обладнання з відповідними характеристиками. Мінімальні вимоги включають багатоядерний процесор Intel Xeon або еквівалент з тактовою частотою не менше 2,5 GHz та більше 4 ядер. Оперативна пам'ять повинна складати хоча б 16 ГБ, а для забезпечення більшої продуктивності – 32 ГБ і більше, особливо при обробці

великих обсягів даних. Що стосується зберігання даних, то мінімальний обсяг жорсткого диска має бути 500 ГБ на SSD для операційної системи, а також рекомендується додатковий диск для зберігання даних (можна використовувати HDD). Рекомендується використовувати SSD обсягом 1 ТБ і більше для покращення швидкості обробки даних.

Для забезпечення високої пропускної здатності мережевого з'єднання, мінімальна швидкість повинна становити 1 Gbit/c, але для великих навантажень оптимальною буде швидкість 10 Gbit/c. Система охолодження повинна бути достатньо ефективною, щоб підтримувати стабільну роботу сервера навіть при великих навантаженнях.

Також необхідно передбачити наявність джерела безперебійного живлення (UPS), щоб уникнути збоїв при відключенні електроенергії, а для забезпечення надійності зберігання даних – підтримку RAID. Сервер має працювати на серверній операційній системі, такій як Windows Server або Linux (наприклад, Ubuntu Server чи CentOS), яка підтримує всі необхідні програмні компоненти, включаючи вебсервери (Apache, Nginx), сервери баз даних (MySQL, PostgreSQL) та інші служби, необхідні для роботи застосунку.

Важливою вимогою є можливість масштабування серверного обладнання, щоб забезпечити підтримку висхідного навантаження без значних втрат продуктивності. Також сервери повинні бути розміщені в спеціалізованому приміщенні з належною фізичною безпекою, щоб доступ до них мали лише уповноважені особи.

4.5 Вимоги до інформаційної та програмної сумісності

Клієнт повинен мати в наявності наступні програмні засоби: останні версії веббраузерів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge або Safari, що підтримують JavaScript, HTML5, CSS3. Операційна система повинна бути Windows (версії 7 і вище), macOS або Linux з актуальними оновленнями безпеки.

Вебзастосунок повинен бути сумісним з реляційними базами даних, такими як MySQL, PostgreSQL або іншими базами даних, що підтримують стандарт SQL

для зберігання та обробки даних. Для забезпечення повної функціональності також можуть використовуватися програми для обробки текстових і табличних документів, такі як Microsoft Office або інші програмні засоби для експорту та імпорту даних у формати Excel, CSV.

Сервер повинен відповідати наступним вимогам до програмної сумісності: По-перше, сервер повинен підтримувати операційні системи, сумісні з Java, зокрема Windows Server, Linux (Ubuntu Server, CentOS, Debian) або інші, що мають підтримку Java Runtime Environment (JRE) та Java Development Kit (JDK). Сервер також має бути оснащений відповідною версією JDK (не нижче Java 11, рекомендовано Java 17 або новіше) для запуску Spring Boot додатків. Крім того, сервер має підтримувати всі залежності та середовище для виконання Spring Boot додатків, що включає інтерпретацію Java класів та залежностей, які використовуються в проєкті. Оскільки в проєкті використовується Spring Security, сервер має бути налаштований на правильну інтеграцію з цією бібліотекою для забезпечення безпеки доступу до ресурсів, а також на підтримку аутентифікації та авторизації користувачів.

Для роботи з базою даних сервер має бути сумісним з MySQL версії 5.7 або новішої, а також мати встановлений MySQL Connector/J або інший драйвер для Java. Використовується сервер для розгортання Spring Boot додатка, та вбудований сервер Tomcat, який йде за замовчуванням у Spring Boot.

Також сервер має підтримувати стандартні мережеві протоколи, такі як HTTP/HTTPS, для забезпечення безпечного обміну даними між клієнтами та сервером. Для коректної роботи необхідно, щоб сервер підтримував бібліотеки для роботи з Spring Boot, зокрема Spring Data для інтеграції з базою даних, Spring Security для захисту застосунку, а також інші залежності, які використовуються у проєкті.

4.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки відсутні.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання відсутні.

4.8 Спеціальні вимоги

Спеціальні вимоги відсутні.

5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання;
– текст програми;
– керівництво користувача;
– опис програми;
– програму і методику випробувань.

6 Техніко-економічні показники

У кваліфікаційній роботі техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Етапи розробки програмного забезпечення наведено в таблиці А.1.

Таблиця А.1 – Етапи розробки програмного забезпечення

Номер	**Назва етапів роботи**	**Терміни виконання**
1	Аналіз практичної задачі інженерії ПЗ	05.04.2025
2	Аналіз вимог до ПЗ	12.04.2025
3	Розробка архітектури ПЗ	19.04.2025
4	Розробка проєкту ПЗ	02.05.2025
5	Реалізація та тестування ПЗ	15.05.2025

8 Порядок контролю і приймання

Контроль здійснюється керівником кваліфікаційної роботи на кожному етапі розробки програмного забезпечення. Приймання здійснюється за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

ДОДАТОК Б
ВИХІДНІ ТЕКСТИ ПРОГРАМНИХ МОДУЛІВ ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ «AGROFUSION»

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
@ComponentScan(basePackages = {"com.example.logicompanyuser.security"})
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private final UserDetailsService userDetailsService;


    public SecurityConfig(@Qualifier("customUserDetailsServiceImpl")
UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                .authorizeRequests()
                .antMatchers(   "/", "/registration", "/api/**",
"/service/receive-cargo").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/auth/login").permitAll()
                .failureUrl("/auth/login?error=true")
                .defaultSuccessUrl("/auth/index",true)
                .and()
                .logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/auth/logout",
"POST"))
                .invalidateHttpSession(true)
                .clearAuthentication(true)
                .deleteCookies("JSESSIONID")
                .logoutSuccessUrl("/auth/login");
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Override
    public void configure(WebSecurity web) {
        web.ignoring().antMatchers("/static/**", "/images/**", "/css/**",
"/js/**", "/json/**");
    }

    @Bean
```

```
    protected PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(12);
    }


    @Bean
    protected DaoAuthenticationProvider daoAuthenticationProvider() {
        DaoAuthenticationProvider daoAuthenticationProvider = new
DaoAuthenticationProvider();
        daoAuthenticationProvider.setPasswordEncoder(passwordEncoder());
        daoAuthenticationProvider.setUserDetailsService(userDetailsService);
        return daoAuthenticationProvider;
    }
}
```

ApiController.java

```
@RestController
@RequestMapping("/api")
public class ApiController {

    private static final Logger logger =
LoggerFactory.getLogger(ApiController.class);

    @Autowired
    private UserService userService;

    @Autowired
    private AddressesService addressesService;

    @Autowired
    private OrderService orderService;

    @Autowired
    private TechnicCharacterCargoService technicCharacterCargoService;

    @GetMapping("/sender/{id}")
    public ResponseEntity<?> getSender(@PathVariable Long id) {
        logger.info("Отримано запит на дані відправника з ID: {}", id);
        try {
            Optional<User> user = userService.getUserById(id);

            if (user.isPresent()) {
                logger.info("Користувача з ID {} знайдено в базі даних", id);
                return ResponseEntity.ok(user.get());
            } else {
                logger.warn("Користувача з ID {} не знайдено в базі даних", id);
                return
ResponseEntity.status(HttpStatus.NOT_FOUND).body("Користувача не знайдено");
            }
        } catch (Exception e) {
            logger.error("Помилка при отриманні даних відправника з ID {}: ", id,
e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Виникла внутрішня помилка сервера");
        }
    }

    @GetMapping("/receiver/{id}")
    public ResponseEntity<?> getReceiver(@PathVariable Long id) {
        logger.info("Отримано запит на дані отримувача з ID: {}", id);
```

```
        return userService.getUserById(id)
                .<ResponseEntity<?>>map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.status(HttpStatus.NOT_FOUND)
                        .body("Користувача не знайдено"));
    }

    @GetMapping("/address/{id}")
    public ResponseEntity<?> getAddress(@PathVariable Long id) {
        logger.info("Отримано запит на адресу з ID: {}", id);
        try {
            Optional<Addresses> address = addressesService.getAddressById(id);
            if (address.isPresent()) {
                logger.info("Знайдено адресу: {}", address.get());
                return ResponseEntity.ok(address.get());
            } else {
                logger.info("Адресу не знайдено");
                return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Адресу не
знайдено");
            }
        } catch (Exception e) {
            logger.error("Помилка при отриманні даних адреси: ", e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Виникла внутрішня помилка сервера");
        }
    }

    @GetMapping("/cargo/{id}")
    public ResponseEntity<?> getCargo(@PathVariable Long id) {
        logger.info("Отримано запит на вантаж з ID: {}", id);
        try {
            Optional<TechnicCharacterCargo> cargo =
technicCharacterCargoService.getCharacterCargoById(id);
            if (cargo.isPresent()) {
                return ResponseEntity.ok(cargo.get());
            } else {
                return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Вантаж не
знайдено");
            }
        } catch (Exception e) {
            logger.error("Помилка при отриманні даних вантажу: ", e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Виникла внутрішня помилка сервера");
        }
    }

    @GetMapping("/order/{orderId}/status-history")
    public List<OrderStatusHistoryDto> getStatusHistory(@PathVariable Long
orderId) {
        return orderService.getStatusHistoryByOrderId(orderId).stream()
                .map(h -> new OrderStatusHistoryDto(
                        h.getOrder().getId(),
                        h.getStatus().name(),
                        h.getChangedAt()))
                .collect(Collectors.toList());
    }



    @GetMapping("/user/{userId}/orders")
    public ResponseEntity<?> getUserOrders(@PathVariable Long userId) {
        logger.info("Отримано запит на замовлення користувача з ID: {}", userId);
        try {
            List<Order> orders = orderService.getOrdersByUserId(userId);
```

```
            logger.info("Знайдено {} замовлень для користувача з ID {}",
orders.size(), userId);
            return ResponseEntity.ok(orders);
        } catch (Exception e) {
            logger.error("Помилка при отриманні замовлень користувача з ID {}: ",
userId, e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Виникла внутрішня помилка сервера");
        }
    }


    @GetMapping("/{userId}/orders-last-status-date")
    public ResponseEntity<?> getUserOrdersLastStatusDate(@PathVariable Long
userId) {
        logger.info(" Запит: отримання дати останнього оновлення статусів
замовлень для користувача ID = {}", userId);
        try {
            logger.debug("➡ Звернення до сервісу:
orderService.getLastStatusDateByUserId({})", userId);
            List<OrderLastStatusDateDto> lastStatusDates =
orderService.getLastStatusDateByUserId(userId);

            if (lastStatusDates == null || lastStatusDates.isEmpty()) {
                logger.warn(" Дата останнього оновлення статусів замовлень не
знайдена для користувача ID = {}", userId);
            } else {
                logger.info(" Знайдено {} дат останнього оновлення статусів для
користувача ID = {}", lastStatusDates.size(), userId);
                for (OrderLastStatusDateDto dto : lastStatusDates) {
                    logger.debug("Order ID = {}, last changedAt = {}",
                            dto.getOrderId(),
                            dto.getLastChangedAt() != null ?
dto.getLastChangedAt().toString() : "NULL");
                }
            }

            return ResponseEntity.ok(lastStatusDates);

        } catch (Exception e) {
            logger.error("Помилка при отриманні дат останнього оновлення статусів
користувача ID = {}: {}", userId, e.getMessage(), e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Виникла внутрішня помилка сервера");
        }
    }


    @PutMapping("/orders/{orderId}/status")
    public ResponseEntity<?> updateOrderStatus(@PathVariable Long orderId,
@RequestBody StatusUpdateRequest request) {
        logger.info("Отримано статус для оновлення в контролері: {}",
request.getNewStatus());
        logger.info("Отриманий статус для оновлення: {}", request.getNewStatus());
        logger.info("Оновлення статусу замовлення ID: {}, новий статус: {}",
orderId, request.getNewStatus());
        try {
            boolean updated = orderService.updateOrderStatus(orderId,
request.getNewStatus());
            if (updated) {
                return ResponseEntity.ok(Collections.singletonMap("message",
"Статус оновлено"));
            } else {
```

```
                return
ResponseEntity.status(HttpStatus.NOT_FOUND).body("Замовлення не знайдено");
            }
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Помилка при
оновленні статусу");
        }
    }

}
```

ApprovalController.java

```
@Controller
public class ApprovalController {

    private final OrderService orderService;
    private final UserRepository userRepository;
    private final ApproverRepository approverRepository;
    private final ApprovalSheetService approvalSheetService;

    @Autowired
    public ApprovalController(OrderService orderService,
                              UserRepository userRepository,
                              ApprovalSheetService approvalSheetService,
                              ApproverRepository approverRepository
                              ) {
        this.orderService = orderService;
        this.userRepository = userRepository;
        this.approvalSheetService = approvalSheetService;
        this.approverRepository = approverRepository;
    }

    @GetMapping("/approvals/{orderId}")
    public String getApprovalPage(@PathVariable Long orderId, Model model,
Principal principal) {
        Optional<Order> orderOptional = orderService.findById(orderId);

        if (orderOptional.isEmpty()) {
            model.addAttribute("error", "Замовлення не знайдено.");
            return "error";
        }

        Order order = orderOptional.get();

        User currentUser = userRepository.findByEmail(principal.getName())
                .orElseThrow(() -> new RuntimeException("Користувача не
знайдено"));

        boolean isApprover = approvalSheetService.isUserApprover(orderId,
currentUser.getId());
        if (!isApprover) {
            model.addAttribute("error", "Ви не маєте прав на підтвердження цього
замовлення.");
            return "error";
        }

        ApprovalSheet approvalSheet = approvalSheetService.findByOrderId(orderId)
                .orElseThrow(() -> new RuntimeException("Лист узгодження не
знайдено"));
```

```
        model.addAttribute("order", order);
        model.addAttribute("user", currentUser);
        model.addAttribute("approvalSheet", approvalSheet);

        return "approval-page";
    }

    @PostMapping("/approvals/action")
    public String handleApprovalAction(@RequestParam Long orderId,
                                       @RequestParam Long approverId,
                                       @RequestParam(required = false) String
action,
                                       @RequestParam(required = false) String
comment,
                                       Principal principal) {

        Approver approver = approverRepository.findById(approverId)
                .orElseThrow(() -> new RuntimeException("Approver not found"));

        if (action == null || action.isEmpty() || action.equals("COMMENT")) {
            approver.setComment(comment);
            approverRepository.save(approver);
            System.out.println("Comment saved");
            return "redirect:/approvals/" + orderId;
        }

        if (approver.getStatus() != AgreementStatus.PENDING) {
            return "redirect:/approvals/" + orderId + "?error=already_decided";
        }

        approver.setStatus(AgreementStatus.valueOf(action));
        approver.setComment(comment);
        approverRepository.save(approver);

                boolean allApproved = approverRepository
                .findByApprovalSheet_Order_Id(orderId).stream()
                .allMatch(a -> a.getStatus() == AgreementStatus.APPROVED);

        if (allApproved) {
            orderService.finalizeOrder(orderId);
        }

        return "redirect:/approvals/" + orderId;
    }
}
```

ReceiverController.java

```
@Controller
@RequestMapping("/service")
public class ReceiverController {

    @Autowired
    private OrderService orderService;

    @Autowired
    private UserService userService;

    @Autowired
    private HistoryService historyService;

    @Autowired
    private AddressesService addressesService;
```

```
    @GetMapping("/receive-cargo")
    public String getMyOrders(Model model, Principal principal) {
        if (principal != null) {
            String userEmail = principal.getName();
            Optional<User> userOptional = userService.getUserByEmail(userEmail);

            if (userOptional.isPresent()) {
                User user = userOptional.get();
                Long userId = user.getId();

                List<Order> orders = orderService.getOrdersByReceiverId(userId);
                model.addAttribute("orders", orders);
            } else {
                model.addAttribute("message", "Користувача не знайдено.");
            }
        } else {
            model.addAttribute("message", "Ви не авторизовані.");
        }
        return "receive-cargo";
    }


    @PostMapping("/receive-cargo")
    public ResponseEntity<String> confirmOrder(@RequestBody OrderConfirmationData
orderConfirmationData) {
        List<String> cells = orderConfirmationData.getCells();
        if (cells == null || cells.size() < 8) {
            return ResponseEntity.badRequest().body("Invalid order data");
        }

        String orderId = cells.get(0);

        Optional<Order> orderOptional = orderService.getOrderById(orderId);
        if (!orderOptional.isPresent()) {
            return ResponseEntity.badRequest().body("Order not found");
        }

        Order order = orderOptional.get();

        orderService.saveOrder(order);

        historyService.saveToHistory(order);

        return ResponseEntity.ok("Cargo received, order status updated, and order
moved to history.");
    }

    @GetMapping("/history-cargo")
    public String getCompletedOrders(Model model, Principal principal) {
        if (principal != null) {
            String userEmail = principal.getName();
            Optional<User> userOptional = userService.getUserByEmail(userEmail);

            if (userOptional.isPresent()) {
                User user = userOptional.get();
                Long userId = user.getId();
                List<Order> doneOrders =
orderService.getOrdersByReceiverIdAndStatus(userId, "DONE");
                model.addAttribute("orders", doneOrders);
            } else {
                model.addAttribute("message", "Користувача не знайдено.");
```

```
            }
        } else {
            model.addAttribute("message", "Ви не авторизовані.");
        }

        return "history-cargo";
    }
    @GetMapping("/status-cargo")
    public String getStatusHistory(Model model, Principal principal) {
        if (principal != null) {
            String userEmail = principal.getName();
            Optional<User> userOptional = userService.getUserByEmail(userEmail);

            if (userOptional.isPresent()) {
                User user = userOptional.get();
                Long userId = user.getId();

                List<OrderStatusHistory> latestStatusHistory =
orderService.getLatestStatusHistoryByUserId(userId);

                model.addAttribute("statusHistory", latestStatusHistory);
            } else {
                model.addAttribute("message", "Користувача не знайдено.");
            }
        } else {
            model.addAttribute("message", "Ви не авторизовані.");
        }
        return "status-cargo";
    }

}
```

SendCargoController.java

```
@Controller
@RequestMapping("/service")
public class SendCargoController {

    private final TechnicCharacterCargoService cargoService;
    private final AddressesService addressesService;
    private final OrderService orderService;
    private final UserRepository userRepository;
    private final ApprovalSheetService approvalSheetService;


    private static final Logger logger =
LoggerFactory.getLogger(SendCargoController.class);

    @Autowired
    public SendCargoController(
            TechnicCharacterCargoService cargoService,
            AddressesService addressesService,
            OrderService orderService,
            UserRepository userRepository,
            ApprovalSheetService approvalSheetService

    ) {
        this.cargoService = cargoService;
        this.addressesService = addressesService;
        this.orderService = orderService;
        this.userRepository = userRepository;
        this.approvalSheetService = approvalSheetService;
```

```
    }

    @GetMapping("/send-cargo")
    public String getSendCargoPage(Model model, Principal principal) {
        CargoForm cargoForm = new CargoForm();
        OrderForm orderForm = new OrderForm();

        model.addAttribute("cargoForm", cargoForm);
        model.addAttribute("orderForm", orderForm);

        Optional<User> currentUserOpt =
userRepository.findByEmail(principal.getName());

        List<User> users;
        if (currentUserOpt.isPresent()) {
            User currentUser = currentUserOpt.get();
            users = userRepository.findAll()
                    .stream()
                    .filter(user -> !user.getId().equals(currentUser.getId()))
                    .collect(Collectors.toList());
        } else {
            users = userRepository.findAll();
        }

        model.addAttribute("users", users);

        return "send-cargo";
    }

    @PostMapping("/send-cargo")
    public String sendCargo(OrderForm orderForm, Principal principal,
RedirectAttributes redirectAttributes, CargoForm cargoForm) {
        String userEmail = principal.getName();
        Optional<User> userOptional = userRepository.findByEmail(userEmail);

        if (userOptional.isPresent()) {
            User user = userOptional.get();
            Long senderId = user.getId();
            logger.info("ID поточного користувача (відправник): {}", senderId);
            orderForm.setClientSenderId(senderId);
            Long receiverId = orderForm.getClientReceiverId();
            logger.info("ID вибраного отримувача з форми: {}", receiverId);
            if (receiverId == null) {
                logger.warn("ID отримувача порожній");
                redirectAttributes.addFlashAttribute("error", "ID отримувача не
може бути порожнім.");
                return "redirect:/service/send-cargo";
            }

            Addresses addresses = cargoForm.getAddresses();
            Addresses savedAddress = addressesService.registerAddresses(
                    addresses.getShippingPoint(),
                    addresses.getIntermediatePoints1(),
                    addresses.getIntermediatePoints2(),
                    addresses.getIntermediatePoints3(),
                    addresses.getDeliveryPoint()
            );

            if (savedAddress == null) {
                logger.error("Помилка при реєстрації адреси");
                redirectAttributes.addFlashAttribute("error", "Помилка при
реєстрації адреси.");
                return "redirect:/service/send-cargo";
```

```
            }

            TechnicCharacterCargo cargo = cargoForm.getCargo();
            TechnicCharacterCargo savedCargo =
cargoService.registerCharacterCargo(
                    cargo.getWeight(),
                    cargo.getExtent(),
                    cargo.getCargoName(),
                    cargo.getQuantity(),
                    cargo.getTypeCargo(),
                    cargo.getTypeCargoDelivery(),
                    cargo.isSpecialConditions(),
                    cargo.getMinTemperature(),
                    cargo.getMaxTemperature(),
                    cargo.getStorageConditions(),
                    cargo.isAiring(),
                    cargo.getMinHumidity(),
                    cargo.getMaxHumidity(),
                    cargo.isPerishableGoods(),
                    cargo.isOversizedCargo()
            );

            orderForm.setAddressId(savedAddress.getId());
            orderForm.setCargoId(savedCargo.getId());

            logger.info("Форма вантажу: " + cargoForm);

            try {
                Long orderId = orderService.saveDraftOrder(orderForm);
                logger.info("Замовлення успішно створено з ідентифікатором: " +
orderId);
                redirectAttributes.addFlashAttribute("success", "Замовлення
успішно створено з ідентифікатором: " + orderId);

                List<Long> approverIds = new ArrayList<>();
                approverIds.add(senderId);
                approverIds.add(receiverId);

                approvalSheetService.createInitialApprovalSheet(orderId,
approverIds);


                String formattedDate =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd.MM.yyyy"));
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy");
                orderForm.setOrderDate(LocalDate.parse(formattedDate, formatter));
                return "redirect:/approvals/" + orderId;

            } catch (IllegalArgumentException e) {
                logger.error("Помилка при створенні замовлення: " +
e.getMessage(), e);
                redirectAttributes.addFlashAttribute("error", e.getMessage());
            } catch (Exception e) {
                logger.error("Неочікувана помилка при створенні замовлення: " +
e.getMessage(), e);
                redirectAttributes.addFlashAttribute("error", "Сталася помилка при
створенні замовлення. Будь ласка, спробуйте ще раз.");
            }

            return "redirect:/service/send-cargo";
        } else {
            logger.warn("Користувача не знайдено: " + userEmail);
```

```
            redirectAttributes.addFlashAttribute("error", "Користувача не
знайдено. Будь ласка, увійдіть до свого облікового запису.");
            return "redirect:/login";
        }
    }
}
```

UserService.java

```
@Service
public class UserService {

    private static final Logger logger =
LoggerFactory.getLogger(UserDetailsServiceImpl.class);


    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    @Autowired
    public UserService(UserRepository userRepository, PasswordEncoder
passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    public Optional<User> getUserById(Long id) {
        return userRepository.findById(id);
    }

    public Optional<User> getUserByEmail(String email) {
        return userRepository.findByEmail(email);
    }

    public Optional<User> getUserByPhoneNumber(String phoneNumber) {
        logger.info("Fetching user with phone number: {}", phoneNumber);
        Optional<User> user = userRepository.findByPhoneNumber(phoneNumber);
        if (user.isPresent()) {
            logger.info("User found: {}", user.get());
        } else {
            logger.info("User with phone number {} not found", phoneNumber);
        }
        return user;
    }

    public User registerUser(String email, String password, String firstName,
String lastName, String patronymic, String phoneNumber,
                             LocalDate birthdayDate, String sex, String education,
String organization, String unit, String position) {
        if (email == null || password == null || firstName == null || lastName ==
null || patronymic == null || phoneNumber == null || birthdayDate == null ||
education == null || organization == null || sex == null || unit == null ||
position == null) {
            throw new NullPointerException("One of the parameters is null");
        }

        if (userRepository.findByEmail(email).isPresent()) {
            throw new IllegalArgumentException("Email already in use");
        }

        User user = new User();
        user.setEmail(email);
```

```
user.setPassword(passwordEncoder.encode(password));
user.setFirstName(firstName);
user.setLastName(lastName);
user.setPatronymic(patronymic);
user.setPhoneNumber(phoneNumber);
user.setBirthdayDate(birthdayDate);
user.setEducation(education);
user.setOrganization(organization);
user.setSex(sex);
user.setUnit(unit);
user.setPosition(position);
user.setRole(Role.USER);
user.setStatus(Status.ACTIVE);
return userRepository.save(user);
```

ДОДАТОК В
ОПИС ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ «AGROFUSION»

1 Функціональне призначення

1.1 Позначення та найменування програми

Позначення програми: AF-LOG-INT-2025

Найменування програми: вебзастосунок для внутрішніх перевезень компанії “Agrofusion”

1.2 Програмне забезпечення, необхідне для функціонування програми

Для коректної роботи вебзастосунка необхідне наступне програмне забезпечення.

Серверна частина (Backend):

– Java Development Kit (JDK) версії 17 або вище;

– Spring Boot Framework (версія v2.3.3.RELEASE);

– Apache Maven – для збірки проєкту;

– MySQL Server (версія 8.0 або вище) □система керування базами даних;

– Spring Security – для реалізації автентифікації та авторизації;

– Hibernate / JPA – для ORM доступу до бази даних.

Клієнтська частина (Frontend): Сучасний веббраузер, що підтримує HTML5, CSS3, JavaScript (Google Chrome, Firefox, Microsoft Edge тощо)

Сервер/Хостинг:

– Apache Tomcat або інший Java-сумісний вебсервер;

– Операційна система: Windows Server, Linux – залежно від розгортання.

1.3 Мови програмування, на яких написана програма

Для реалізації вебзастосунка було використано такі мови програмування:

Java – основна мова серверної частини програми (backend). Застосовується для обробки бізнес-логіки, роботи з базою даних, обробки запитів REST API тощо.

JavaScript – використовується на стороні клієнта (frontend) для реалізації інтерактивності та логіки користувацького інтерфейсу.

HTML (HyperText Markup Language) – мова розмітки, що забезпечує структуру вебсторінок.

CSS (Cascading Style Sheets) – мова стилів, яка використовується для оформлення зовнішнього вигляду елементів інтерфейсу.

2 Функціональне призначення

2.1 Призначення програми

Призначенням програмного забезпечення є автоматизація операції зі створення заявки на перевезення вантажів між внутрішніми підрозділами компанії, генерація документів до цих заяв, облік та звітність.

2.2 Відомості про функціональні обмеження при використанні

Під час використання вебзастосунка для внутрішніх перевезень компанії “Agrofusion” слід враховувати наступні функціональні обмеження:

– Доступ до функціоналу системи надається лише зареєстрованим користувачам.

– Програма функціонує лише за наявності стабільного підключення до внутрішньої корпоративної мережі або інтернету (у разі розміщення на віддаленому сервері).

– Коректна робота вебінтерфейсу гарантована лише в сучасних браузерах, які підтримують HTML5, CSS3 і JavaScript. У застарілих версіях можливе некоректне відображення або порушення функціоналу.

– Вебзастосунок не підтримує одночасне редагування одного й того ж замовлення кількома користувачами. Це може призвести до втрати або конфлікту даних.

– Продуктивність залежить від потужності сервера БД. За великої кількості одночасних запитів можливе зниження швидкодії.

– Алгоритм затвердження замовлень реалізований відповідно до бізнес-процесів компанії і не змінюється без втручання розробника.

3 Функціональне призначення

3.1 Алгоритм програми

Програма працює за принципом клієнт-серверної архітектури та реалізує наступну логіку:

Авторизація користувача:

– Користувач вводить облікові дані.

– Вебзастосунок перевіряє правильність даних за допомогою Spring Security.

– Після успішної автентифікації користувач перенаправляється до інтерфейсу.

Створення нового замовлення:

– Користувач заповнює форму із зазначенням вантажу, адрес відправника та одержувача.

– Дані надсилаються на сервер через POST-запит.

– На сервері створюється об’єкт замовлення, пов’язаний з вантажем, адресами та користувачем.

– Замовлення зберігається в базі даних зі статусом "Чернетка"

Узгодження замовлення:

– Замовлення автоматично передається на узгодження відповідним особам.

– Кожен узгоджувач має можливість схвалити або відхилити замовлення.

– Після остаточного схвалення у випадку ухвалення заяви всіма відповідальними сторонами заява прийнята до роботи. Якщо одна із відповідальних сторін відхилить заяву то заяву буде відхилено.

Отримання вантажу:

– Відповідальна особа відкриває список замовлень зі статусом "Очікує отримання".

– Після фактичного отримання вантажу статус замовлення змінюється на "Отримано" (самим отримувачем).

Історія змін:

– Кожна зміна статусу замовлення фіксується в історії.

– Користувачі можуть переглядати хронологію змін статусів у відповідному інтерфейсі.

Додаткові функції:

– Фільтрація, пошук замовлень, сортування за статусом і датою.

– Обмеження доступу до окремих дій відповідно до ролі користувача.

– Блокування доступу до акаунта користувачу.

3.2 Методи, що використовуються

У вебзастосунка для внутрішніх перевезень компанії “Agrofusion” застосовано такі методи:

– Об’єктноорієнтоване програмування (ООП). Реалізація логіки за допомогою класів, об'єктів, наслідування, інкапсуляції та поліморфізму, що забезпечує гнучкість і масштабованість архітектури.

– Розробка RESTful-сервісів. Комунікація між клієнтом і сервером реалізована через REST API з використанням HTTP-методів (GET, POST), що дозволяє легко обробляти запити та відповіді.

– Інкапсуляція бізнес-логіки в сервісні методи. Ключова логіка роботи із замовленнями, статусами, ТТН та історією реалізована в окремих сервісних класах для спрощення підтримки і тестування.

– Метод логування змін стану. Всі зміни статусу замовлення фіксуються у відповідній таблиці історії, що забезпечує прозорість бізнес-процесу.

– Метод фільтрації та пошуку. Для зручності роботи користувачів реалізовані методи фільтрації замовлень за статусом та періодом дат.

3.3 Структура програми з описом функцій складових частин і зв’язку між ними

Вебзастосунок для внутрішніх перевезень компанії “Agrofusion” складається з трьох основних компонентів:

Клієнтська частина (Frontend)

Функції:

– Забезпечує користувацький інтерфейс для взаємодії з системою.

– Відображає форми для створення, редагування і перегляду замовлень.

– Забезпечує фільтрацію, пошук і сортування замовлень.

– Проводить початкову валідацію введених даних.

– Відправляє HTTP-запити до серверної частини та приймає відповіді.

Технології: HTML, CSS, JavaScript (можливо React.js або Thymeleaf).

Серверна частина (Backend)

Функції:

– Обробляє запити від клієнта.

– Реалізує бізнес-логіку роботи з замовленнями, узгодженням, створенням документів на основі вхідної інформації.

– Виконує автентифікацію та авторизацію користувачів.

– Працює з базою даних через ORM (Hibernate).

– Генерує звіти.

– Реалізує логіку збереження та оновлення історії змін статусів.

Технології: Java, Spring Boot, Spring Security, Hibernate.

Система управління базою даних (СУБД)

Функції:

– Зберігає інформацію про користувачів, замовлення, вантажі, адреси, історію змін.

– Забезпечує цілісність і доступність даних.

Технології: MySQL.

Взаємозв’язки між складовими частинами:

Вебзастосунок для внутрішніх перевезень компанії “Agrofusion” складається з трьох основних компонентів: клієнтської частини, серверної частини та системи управління базою даних.

3.3 Зв’язок програми з іншими програмами

На момент розробки вебзастосунок для внутрішніх перевезень компанії “Agrofusion” не інтегрується з іншими програмними продуктами. Програма є автономною та не потребує обміну даними з зовнішніми системами або сервісами для забезпечення своєї функціональності.

4 Необхідні технічні засоби для вебзастосунка для внутрішніх перевезень компанії “Agrofusion

Для ефективної роботи вебзастосунка необхідні такі технічні засоби:

Комп’ютери та ноутбуки:

– Операційні системи: Windows, macOS, Linux

– Мінімальні характеристики:

– Процесор: Intel Core i3 або еквівалент

– ОЗП: від 4 ГБ (рекомендовано 8 ГБ)

– Вільне місце на диску: від 2 ГБ

– Сумісність з браузерами: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari

Інтернет-з’єднання:

– Мінімальна швидкість: 1 Мбіт/с

– Рекомендована швидкість: 5 Мбіт/с і вище

– Можливість роботи через локальну мережу

Монітори:

– Мінімальна роздільна здатність: 1280×1024

– Рекомендована діагональ: від 19 дюймів

Пристрої введення:

– Стандартна клавіатура та миша

– Підтримка сенсорних екранів (опціонально)

Засоби захисту:

– Антивірусне програмне забезпечення

– Налаштовані засоби безпеки для захисту від несанкціонованого доступу

5 Виклик та завантаження вебзастосунка для внутрішніх перевезень компанії “Agrofusion

Вебзастосунок встановлено на локальному сервері підприємства. Доступ до нього здійснюється через браузер за внутрішньою мережею шляхом переходу за локальною URL-адресою: http://localhost:8080/auth/login

Програма запускається автоматично при зверненні до цієї адреси. Встановлення на клієнтські пристрої не вимагається, оскільки вся обробка даних відбувається на сервері, а користувач працює через інтерфейс у веббраузері.

Перед початком роботи користувач проходить автентифікацію через форму входу.

6 Вхідні дані вебзастосунка для внутрішніх перевезень компанії Agrofusion

6.1 Характер, організація і попередня підготовка вхідних даних

Вхідні дані вводяться користувачами вручну через інтерфейс вебзастосунка або формуються автоматично на основі вибраних параметрів. Дані стосуються інформації про замовлення перевезення, вантаж, адреси відправлення та доставлення, відповідальних осіб, а також супровідних документів.

6.2 Формат, описання та спосіб кодування вхідних даних

Формат введення: текстові поля, числові значення, дати, розкривні списки, прапорці, завантаження файлів.

Основні типи даних:

– Ідентифікатори користувачів та підрозділів – цілі числа (ID);

– Дата та час – формат YYYY-MM-DD HH:mm;

– Адреси – рядки тексту (UTF-8);

– Інформація про вантаж – числові та текстові значення;

– Документи – файли у форматах (.PDF, .XLSX, .CSV);

– Кодування: усі текстові дані зберігаються у форматі UTF-8, що забезпечує підтримку української мови.

Дані проходять валідацію як на клієнтській, так і на серверній стороні для забезпечення коректності введення та цілісності інформації.

7 Вихідні дані вебзастосунка для внутрішніх перевезень компанії Agrofusion

7.1 Характер, організація і попередня підготовка вихідних даних

Вихідні дані формуються як результат обробки введених користувачами заявок, інформації про вантажі, маршрути, статуси перевезень тощо. Дані виводяться у вигляді таблиць, звітів, журналів змін статусів та сформованих документів (наприклад, товарно-транспортних накладних). Користувачі отримують інформацію через інтерфейс вебзастосунка або мають змогу завантажити результати у вигляді PDF-файлів.

7.2 Формат, описання та спосіб кодування вихідних даних

Вебінтерфейс: табличні дані, повідомлення, статуси, динамічні форми

– Документи – файли у форматах (.PDF, .XLSX, .CSV);

– Статуси замовлень (рядкові значення: «Очікує», «Затверджено», «Виконано» тощо);

– Журнал змін статусів (дата, час, користувач, новий статус);

– Звітні дані про замовлення;

– Кодування: усі текстові дані зберігаються у форматі UTF-8, що забезпечує підтримку української мови.

ДОДАТОК Г
ІНСТРУКЦІЯ КОРИСТУВАЧА (ОПЕРАТОРА) ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ «AGROFUSION»

1 Призначення програми

1.1 Функціональне призначення програми

Вебзастосунок для внутрішніх логістичних перевезень компанії “Agrofusion” створено для автоматизації, оптимізації та контролю всього процесу транспортування вантажів між підрозділами. Його основна мета – забезпечити зручне управління заявками: створення, узгодження, моніторинг і завершення логістичних операцій.

Серед функцій: оформлення заявок із деталями вантажу, адресами та умовами доставлення; колегіальне узгодження через інтерактивний лист; відстеження статусів із повною історією змін; підтвердження доставлення з можливістю друку документів. Доступна аналітика зі звітами, фільтрами, графіками та експортом у різні формати. Інтерфейс повністю адаптований до мобільних пристроїв.

1.2 Експлуатаційне призначення програми

Вебзастосунок призначений для щоденного використання логістичними підрозділами компанії “Agrofusion” через внутрішню мережу або захищений доступ. Його мета – забезпечити прозорість і контроль логістичного процесу, від створення заявки до підтвердження доставлення, спростити взаємодію між підрозділами та зменшити паперовий документообіг завдяки автоматичному формуванню PDF-документів.

Вебзастосунок також дозволяє аналізувати ефективність логістики в реальному часі. Застосунок використовують логісти для обробки заявок, узгоджувачі – для схвалення, а адміністратор – для управління обліковими записами та техпідтримки.

1.3 Склад функцій

Вебзастосунок включає такі функціональні блоки.

«Відправити вантаж» – забезпечує заповнення заявки з адресами складів відправлення і доставлення, вибором отримувача, введенням параметрів вантажу (вага, об’єм, кількість), типу вантажу й доставлення, а також спеціальних умов транспортування (температурний режим, вологість, вентиляція). Користувач може подати або скасувати заявку.

«Отримати вантаж» – дозволяє переглядати замовлення для приймання, фільтрувати їх за статусом, переглядати деталі заявки, змінювати статус доставлення, підтверджувати отримання вантажу та друкувати документи (зокрема товарно-транспортну накладну).

«Статус вантажу» – дає змогу переглядати список усіх замовлень із фільтрацією, бачити поточний статус і історію змін у зручному модальному вікні.

«Історія замовлень» – містить завершені або відхилені заявки з доступом до друку детальної інформації та ТТН, підтримує фільтрацію за датами та статусами.

«Звіти» – відображає статистичні картки (кількість замовлень, вантажів, маршрутів тощо), дозволяє формувати звіти за різними критеріями, переглядати їх у вигляді графіків або таблиць і експортувати у формати CSV, XLSX, PDF, JSON, XML, TXT.

«Авторизація та вихід» – відповідає за вхід до системи за логіном і паролем та безпечний вихід користувача.

2 Умови виконання програми

2.1 Мінімальний склад апаратних засобів

Для роботи вебзастосунка достатньо стандартного офісного обладнання. Мінімальні вимоги – ПК, ноутбук або мобільний пристрій (смартфон, планшет) з Інтернетом. Рекомендовані технічні характеристики: процесор від 1.5 ГГц,

оперативна пам’ять не менше 4 ГБ, дисплей з розширенням мінімум 1366×768 пікселів. Для друку документів (товарно-транспортна накладна, підтвердження отримання) бажано мати підключення до принтера — фізичного або віртуального.

2.2 Мінімальний склад програмних засобів

Для користування вебзастосунком не потрібно встановлювати додаткове програмне забезпечення, окрім сучасного веббраузера. Мінімальні вимоги до програмного забезпечення включають операційну систему Windows 10 або новішу, Linux, macOS, а також мобільні операційні системи Android 10+ або iOS 13+. Рекомендовані браузери – Google Chrome, Mozilla Firefox, Microsoft Edge або Safari, з підтримкою JavaScript і CSS3. Необхідна стабільна робота Інтернет-з’єднання зі швидкістю не менше 1 Мбіт/с. У браузері має бути увімкнена підтримка cookies та local storage. Для перегляду PDF-документів достатньо стандартного переглядача, вбудованого в браузер.

2.3 Вимоги до персоналу (користувачів)

Користувачами вебзастосунка є співробітники логістичних підрозділів компанії і системні адміністратори. Також необхідне розуміння структури внутрішніх логістичних процесів, уміння заповнювати форми заявок та уважно перевіряти внесену інформацію, вміти аналізувати зміст замовлень, приймати обґрунтовані рішення та залишати коментарі. Адміністратор має володіти навичками керування обліковими записами та базовими знаннями у сфері підтримки інформаційних систем.

3 Виконання програми

3.1 Завантаження та запуск програми

Для того, щоб користуватися вебзастосунком потрібно пройти процедуру реєстрації. За реєстрацію нових користувачів відповідальний адміністратор системи. Після процедури реєстрації користувач має пройти авторизацію. На

рисунку Г.1 представлено сторінку авторизації користувачів вебзастосунка для внутрішніх перевезень компанії “Agrofusion”.

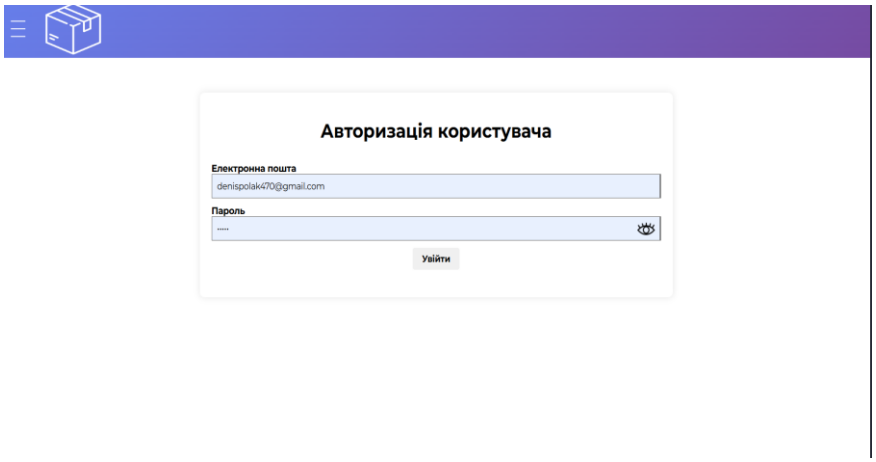


Рисунок Г.1 – Сторінка авторизації користувачів вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

Після авторизації, перша сторінка яка зустрічає нового користувача – це головна сторінка вебзастосунка (рисунок Г.2).

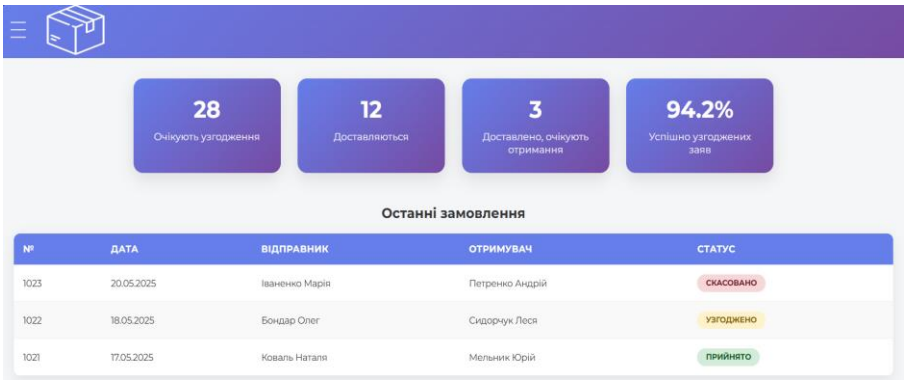


Рисунок Г.2 – Головна сторінка вебзастосунка для внутрішніх перевезень компанії “Agrofusion”

На головній сторінці розташована основна оперативна інформація пов’язана з поточним користувачем, це дозволяє швидко оцінювати ситуацію по заявкам. Для вибору операції потрібно відкрити навігаційне меню, яке обведено червоним на рисунку Г.3.

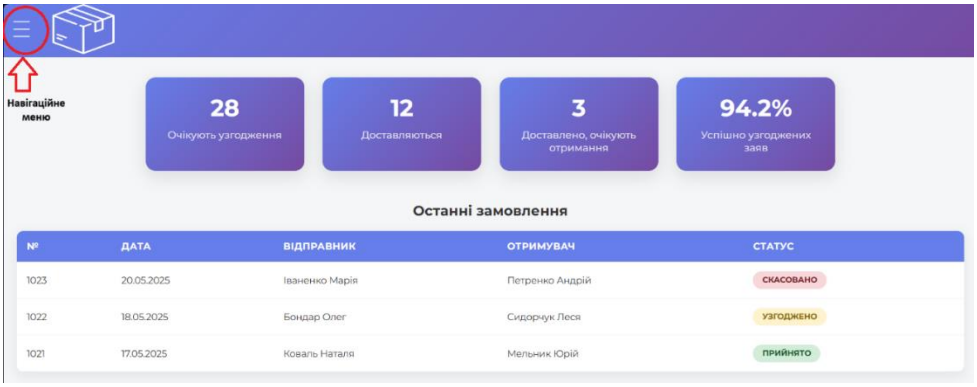


Рисунок Г.3 – Розташування навігаційного меню

Після відкриття навігаційного меню користувач може обрати одну з доступних операцій, щоб перейти на нову сторінку з відповідним функціоналом (рисунок Г.4).

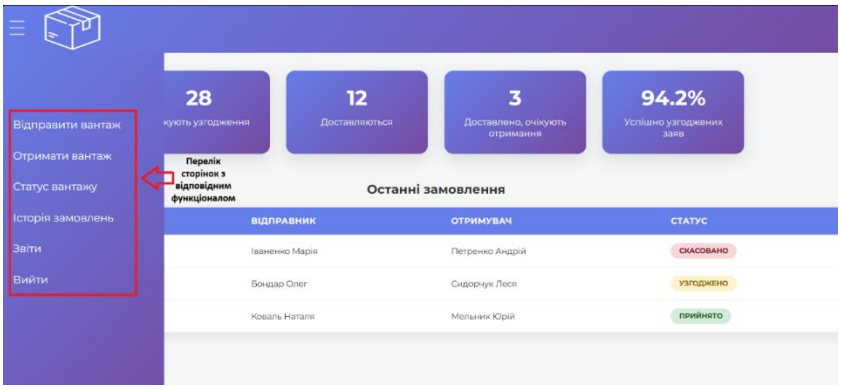


Рисунок Г.4 – Перелік сторінок з відповідним функціоналом

Далі буде описано функціональне призначення кожної сторінки вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion”.

Відправити вантаж – створення заявки з даними про адреси, отримувача, вантаж (вага, об’єм, кількість, тип, умови). Можливість надіслати або скасувати заявку.

Отримати вантаж – перегляд, обробка та підтвердження отримання вантажів між підрозділами, інтерфейс для логістів підрозділу, що приймають вантаж.

Статус вантажу – список замовлень з фільтрацією за статусом, перегляд деталей і історії змін статусу у модальному вікні.

Історія замовлень – перегляд завершених і відхилених заявок з фільтрами за статусом і датами, детальна інформація про замовлення.

Звіти – створення та перегляд аналітичних звітів зі статистичними картками, фільтрами за періодами й типами аналізу, інтерактивними графіками та таблицями, експортом у CSV. Адаптивний інтерфейс і клавіатурні скорочення для зручності.

Вийти – вихід з акаунта з переходом на сторінку авторизації.

3.2 Виконання програми

Детальний опис процесу відправлення вантажу.

На рисунку Г.5 представлено сторінку “Відправити вантаж”

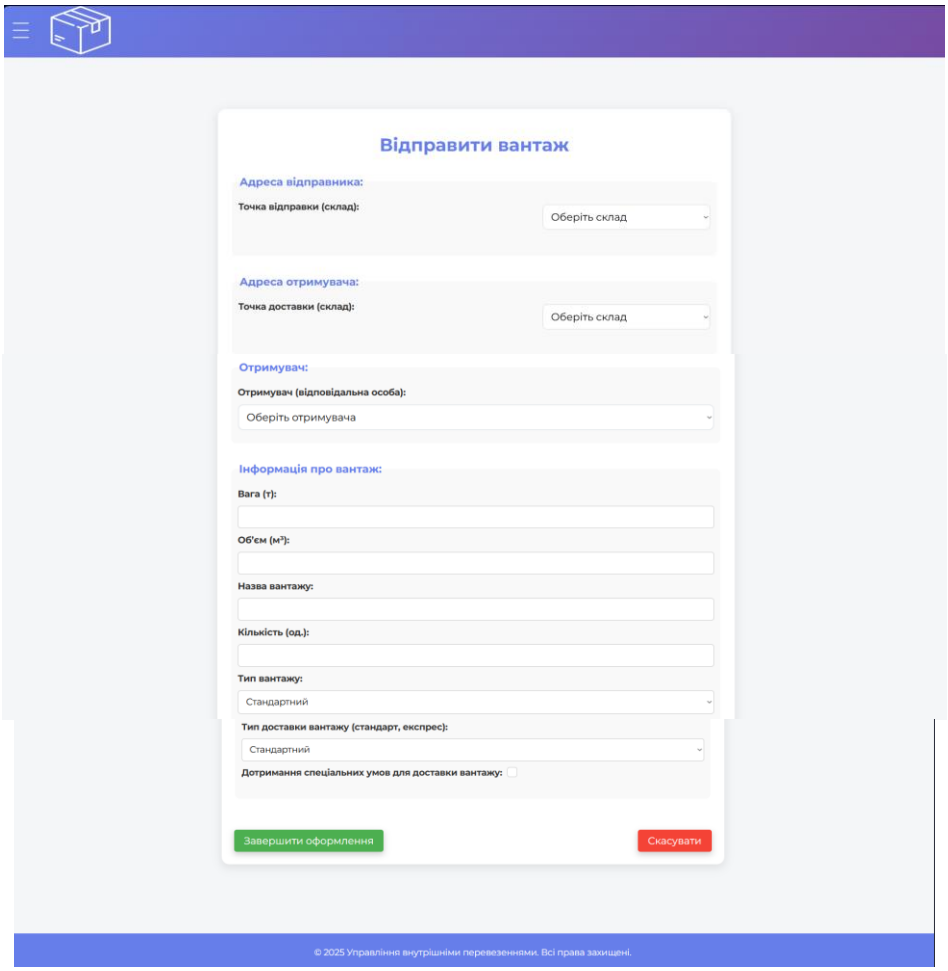


Рисунок Г.5 – Сторінка “Відправити вантаж”

Визначення адрес відправлення та доставлення:

Обирається точка відправлення зі списку доступних складів (рисунок Г.6).

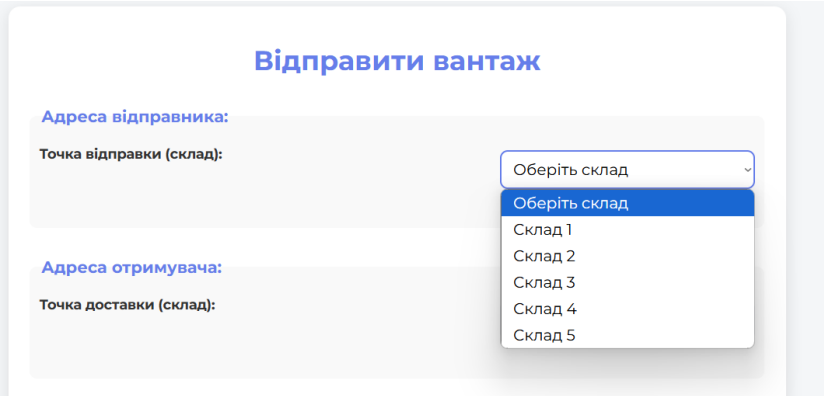


Рисунок Г.6 – Вибір адреси відправлення

Вибір точки доставлення з того ж переліку складів Можливість доставлення між будь-якими складами мережі (точка відправлення і доставлення не мають збігатися!).

Визначення отримувача: у полі "Отримувач (відповідальна особа)" необхідно обрати конкретну особу з розкривного списку, який містить усіх зареєстрованих у системі користувачів. Кожен користувач відображається у форматі "Прізвище Ім'я По батькові", що дозволяє легко знайти потрібну особу серед інших користувачів системи. (рисунок Г.7)

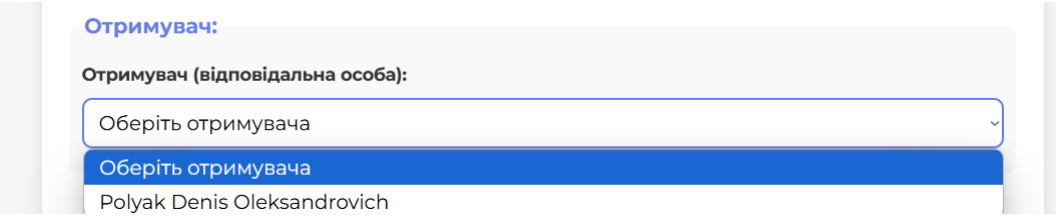


Рисунок Г.7 – Вибір отримувача

Вибір отримувача є обов'язковим для завершення оформлення відправлення. Обрана особа стане основним контактом для всіх питань щодо цього відправлення, включаючи повідомлення про готовність вантажу до отримання та узгодження

деталей доставлення. Переконайтеся, що обираєте правильного отримувача, оскільки саме ця особа буде відповідальною за прийняття вантажу та підтвердження його отримання в повному обсязі відповідно до заявлених характеристик.

Характеристики вантажу: у полі "Вага" введіть загальну вагу вантажу в тоннах. Вебзастосунок вимагає вказати мінімум 1 тонну, тому якщо ваш вантаж важить менше, округліть до найближчого цілого числа або зверніться до адміністратора системи для уточнення процедури обробки легких вантажів (рисунок Г.8).

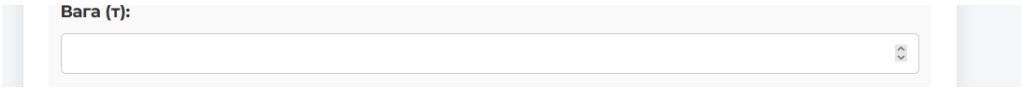


Рисунок Г.8 – Поле “Вага”

Поле "Об'єм" потребує вказання загального об'єму вантажу в кубічних метрах. Мінімальне значення також становить 1 м³. Для правильного розрахунку об'єму виміряйте довжину, ширину та висоту всього вантажу і перемножте ці показники (рисунок Г.9).

Рисунок Г.9 – Поле “Об’єм”

В полі "Назва вантажу" вкажіть стислу, але зрозумілу назву того, що відправляєте (рисунок Г.10).

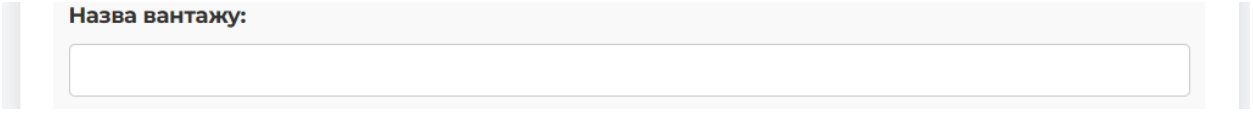


Рисунок Г.10 – Поле “Назва вантажу”

Поле "Кількість" призначене для вказання загальної кількості окремих одиниць вантажу (рисунок Г.11).

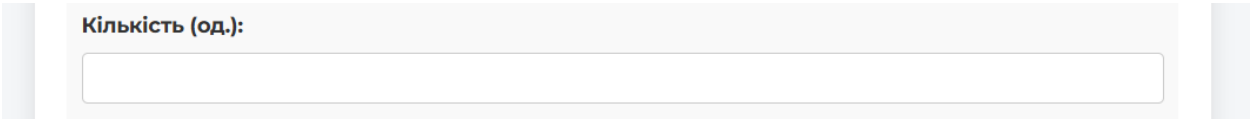


Рисунок Г.11 – Поле “Кількість”

Оберіть тип вантажу відповідно до його характеристик. "Стандартний" підходить для звичайних товарів, які не потребують особливих умов поводження. "Крихкий" слід обирати для товарів, які можуть пошкодитися при необережному поводженні. Вибір правильного типу вантажу забезпечує відповідне поводження з ним під час завантаження, транспортування та розвантаження (рисунок Г.12).

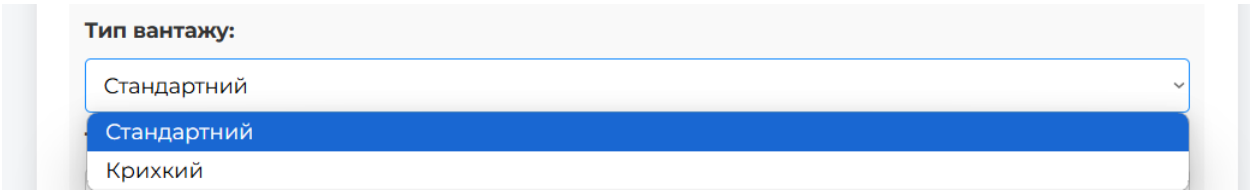


Рисунок Г.12 – Поле “Тип вантажу”

Виберіть швидкість доставлення відповідно до ваших потреб. "Стандартний" тип доставлення використовується для звичайних відправлень без особливих вимог до термінів. "Експрес" варто обирати, коли вантаж потрібно доставити якнайшвидше, але зверніть увагу, що це може вплинути на вартість послуги (рисунок Г.13).

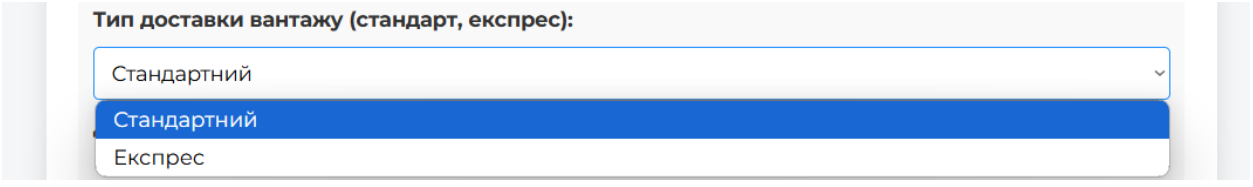


Рисунок Г.13 – Поле “Тип доставлення вантажу”

Спеціальні умови транспортування: якщо ваш вантаж потребує особливих умов під час перевезення або зберігання, поставте галочку в полі "Дотримання спеціальних умов для доставлення вантажу". Після цього з'явиться додаткове меню з різними опціями, з яких ви зможете обрати потрібні для вашого конкретного випадку (рисунок Г.14).

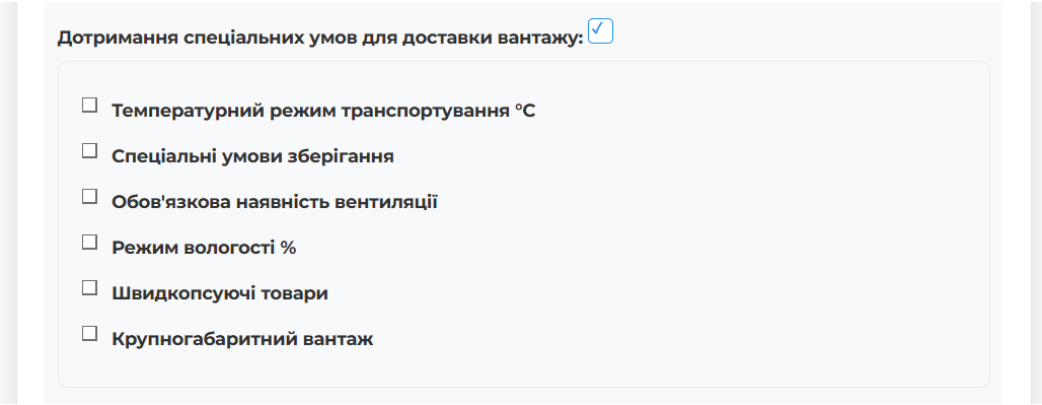


Рисунок Г.14 – Меню “Дотримання спеціальних умов для доставлення вантажу”

Температурний режим транспортування: якщо ваш вантаж чутливий до температури, поставте галочку навпроти "Температурний режим транспортування °C". З'являться два поля для вказання мінімальної та максимальної допустимої температури. Ви можете встановити значення від -50°C до +50°C. Заповніть обидва поля, навіть якщо у вас є лише одне обмеження - в такому випадку в другому полі вкажіть крайнє припустиме значення (рисунок Г.15).

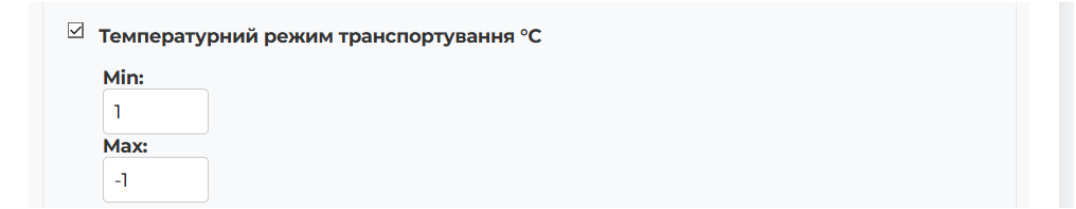


Рисунок Г.15 – Поля “Температурний режим транспортування”

Умови зберігання та вентиляції: поставте галочку навпроти "Спеціальні умови зберігання", якщо ваш вантаж потребує особливого поводження на складах.

Опція "Обов'язкова наявність вентиляції" призначена для вантажів, яким потрібен постійний повітрообмін під час транспортування (рисунок Г.16).

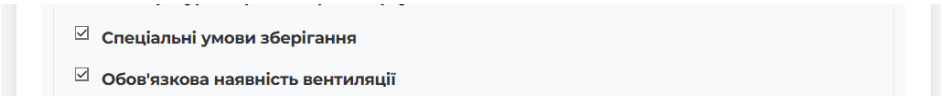


Рисунок Г.16 – Чек-бокси “Спеціальні умови зберігання ”

Режим вологості: для товарів, чутливих до вологості, поставте галочку навпроти "Режим вологості %". Вкажіть мінімальний та максимальний допустимі рівні вологості у відсотках від 0% до 100% (рисунок Г.17).

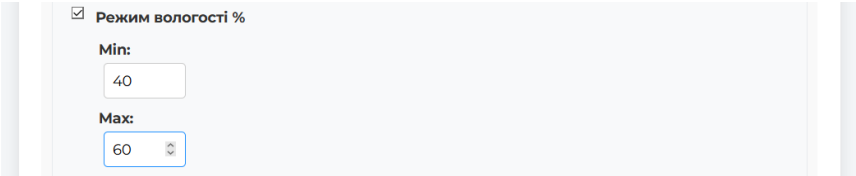


Рисунок Г.17 – Поля “Режим вологості”

Спеціальні категорії вантажів: поставте галочку " Швидкопсувні товари", якщо відправляєте продукти з обмеженим терміном придатності або які можуть швидко втратити свої властивості (рисунок Г.18).

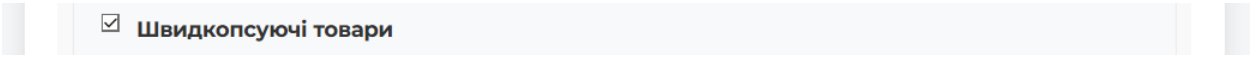


Рисунок Г.18 – Чек-бокс “ Швидкопсувні товари”

Опція " Різногабаритний вантаж" призначена для товарів, які через свої розміри не поміщаються в стандартні вантажні відсіки або потребують спеціального навантажувального обладнання. Поставте цю галочку, якщо довжина, ширина або висота вашого вантажу перевищує звичайні параметри (рисунок Г.19).

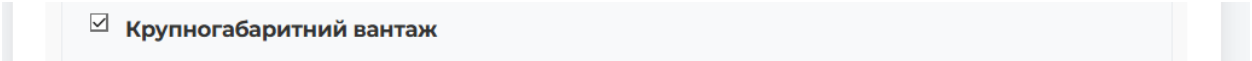


Рисунок Г.19 – Чек-бокс “ Різногабаритний вантаж”

Перевірка введених даних: перед натисканням кнопки "Завершити оформлення" уважно перевірте всю введену інформацію. Вебзастосунок автоматично контролює заповнення обов'язкових полів - якщо якесь поле залишиться порожнім, з'явиться повідомлення про помилку (рисунок Г.20).

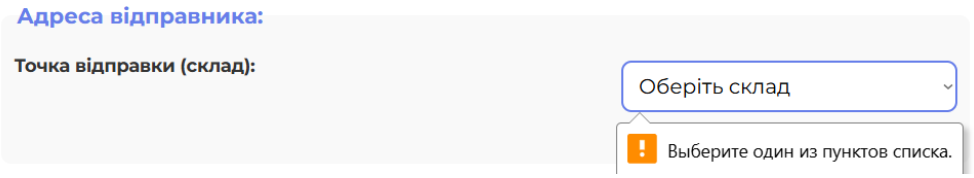


Рисунок Г.20 – Перевірка на заповнення поля “Точка відправлення”

Завершення оформлення: після перевірки всіх даних натисніть зелену кнопку "Завершити оформлення". Вебзастосунок обробить вашу заявку і створить відправлення з унікальним номером для відстеження. Якщо під час оформлення виникне помилка, вебзастосунок покаже детальне повідомлення про те, що саме потрібно виправити. Внесіть необхідні зміни та спробуйте знову. Після чого ви перейдете до наступного етапу, а саме узгодження заяви.

Скасування оформлення: якщо ви передумали або хочете почати заново, натисніть червону кнопку "Скасувати". Це поверне вас до початкової форми з порожніми полями, але всі введені дані будуть втрачені без можливості відновлення.

Детальний опис процесу узгодження замовлення.

Лист узгодження замовлення є важливим документом у системі, який забезпечує контроль та затвердження логістичних операцій через систему колегіального узгодження. Цей документ дозволяє відповідальним особам переглядати, коментувати та приймати рішення щодо конкретних замовлень перед їх остаточним виконанням.

Структура листа узгодження: лист узгодження складається з кількох ключових елементів, які допомагають швидко зорієнтуватися в інформації про замовлення. У верхній частині документа розташовано заголовок з унікальним

номером замовлення та датою його створення. Ця інформація дозволяє однозначно ідентифікувати документ та пов'язати його з конкретним логістичним замовленням у системі. Основну частину документа займає таблиця з переліком усіх узгоджувачів, їхніми статусами, коментарями та можливими діями (рисунок Г.21).

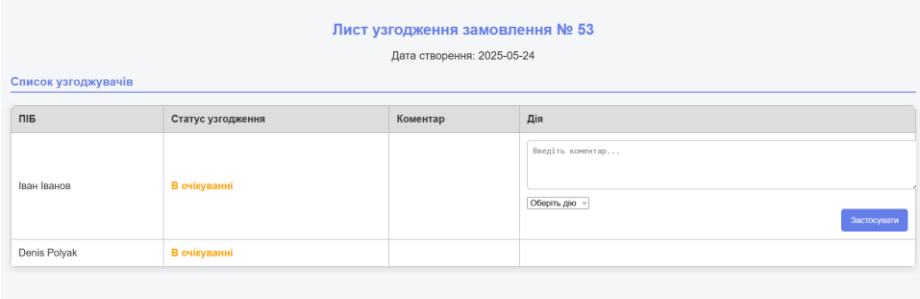


Рисунок Г.21 – Документ “Лист узгодження”

Розуміння статусів узгодження: вебзастосунок використовує три основні статуси для кожного узгоджувача, які відображаються різними кольорами для швидкого візуального сприйняття. Статус "В очікуванні" позначається помаранчевим кольором та означає, що узгоджувач ще не прийняв рішення щодо замовлення і потребує його розгляду. Статус "Узгоджено" відображається зеленим кольором та вказує на те, що узгоджувач схвалив замовлення і дозволяє його подальше виконання. Статус "Відхилено" показується червоним кольором та означає, що узгоджувач не схвалив замовлення з певних причин, які зазвичай описуються в коментарі.

Процес узгодження для активного користувача: якщо ви є одним з узгоджувачів замовлення і ваш статус знаходиться в стані "В очікуванні", ви побачите активну форму в колонці "Дія" навпроти вашого прізвища. Ця форма містить текстове поле для введення коментаря, випадний список для вибору дії та кнопку "Застосувати" для підтвердження вашого рішення. Перед прийняттям рішення уважно ознайомтеся з деталями замовлення та оцініть всі аспекти логістичної операції (рисунок Г.22).

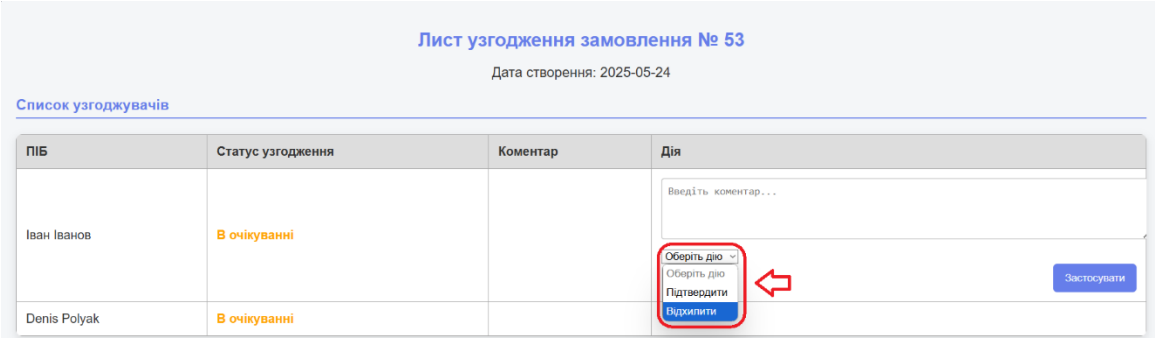


Рисунок Г.22 – Вибір статусу для узгодження заяви

Додавання коментарів до рішення: текстове поле для коментарів дозволяє вам залишити детальне пояснення вашого рішення, що особливо важливо у випадку відхилення замовлення. Коментарі зберігаються в системі та будуть видимі всім учасникам процесу узгодження (рисунок Г.23).

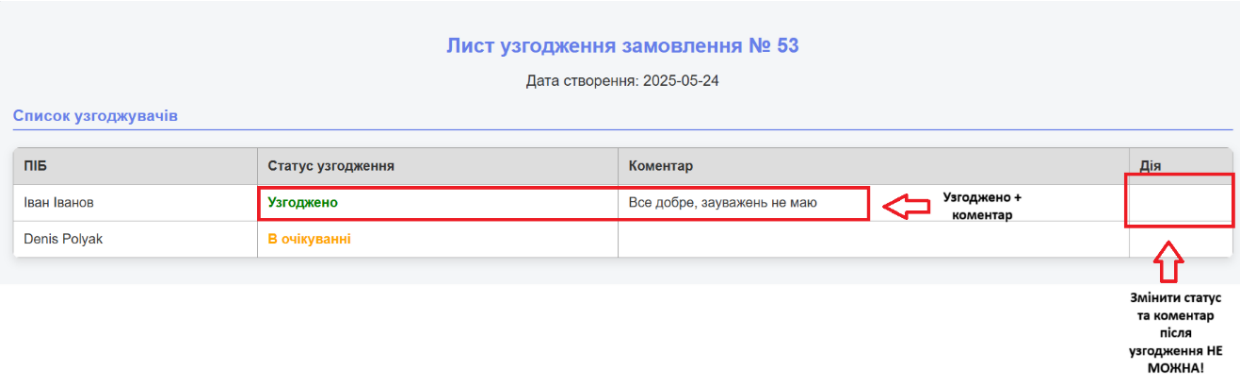


Рисунок Г.23 – Результат успішного узгодження одною із відповідальних осіб

Прийняття рішення про узгодження: для прийняття рішення виберіть відповідну опцію з випадного списку: "Підтвердити" для схвалення замовлення або "Відхилити" для його відхилення. Після введення коментаря (якщо необхідно) та вибору дії натисніть кнопку "Застосувати". Вебзастосунок автоматично збереже ваше рішення, оновить статус у таблиці та повідомить інших учасників процесу про зміну статусу узгодження. Після підтвердження рішення ви більше не зможете його змінити через цю форму (рисунок Г.24).

Перегляд рішень інших узгоджувачів: у таблиці ви можете переглядати статуси та коментарі всіх інших узгоджувачів замовлення. Якщо хтось з

узгоджувачів відхилив замовлення, обов'язково ознайомтеся з їхнім коментарем, щоб зрозуміти причини відхилення та можливі шляхи розв'язання проблеми (рисунок Г.25).

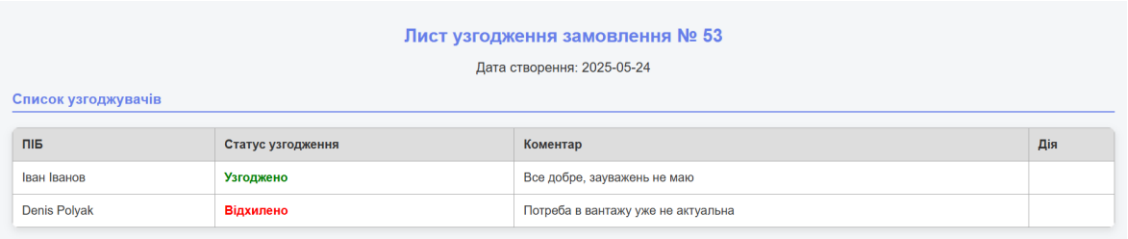


Рисунок Г.24 – Перегляд статусу та коментарів інших осіб

Друк документа узгодження: лист узгодження можна роздрукувати для архівування або передачі в паперовому вигляді. При друку всі кнопки та інтерактивні елементи автоматично приховуються, залишаючи тільки інформаційну частину документа. Це забезпечує чистий та професійний вигляд роздрукованого документа, придатного для офіційного документообігу (рисунок Г.25).

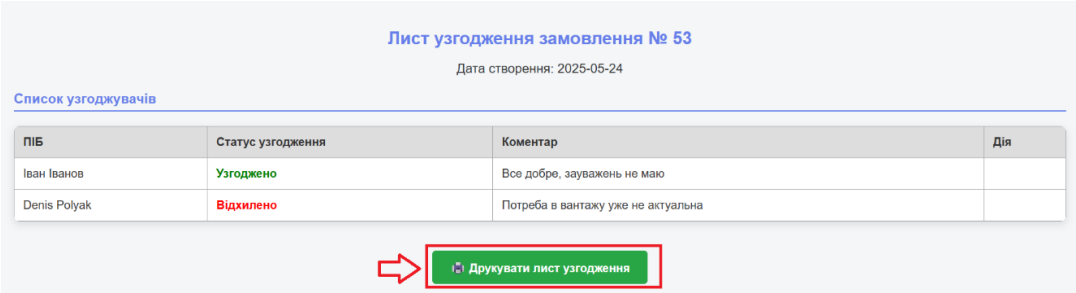


Рисунок Г.25 – Друк документа “Лист узгодження”

Важливі моменти при роботі з узгодженням: завжди уважно перевіряйте всі деталі замовлення перед прийняттям рішення, включаючи номер замовлення, дату створення та статуси інших узгоджувачів. Якщо у вас виникають сумніви щодо замовлення, не соромтеся зв'язатися з ініціатором замовлення або іншими учасниками процесу для отримання додаткової інформації.

Технічні аспекти роботи системи: всі зміни статусів та коментарі зберігаються в реальному часі та синхронізуються між всіма користувачами системи. Якщо у вас виникають технічні проблеми з відправкою форми або збереженням рішення, спробуйте оновити сторінку та повторити дію. У разі постійних технічних проблем зверніться до служби технічної підтримки.

Детальний опис отримання вантажу

На головній сторінці ви побачите таблицю з усіма доступними для отримання замовленнями. Кожен рядок таблиці містить основну інформацію про замовлення: номер ордера, дату створення, поточний статус доставлення, дані відправника та отримувача, маршрут доставлення та інформацію про вантаж (рисунок Г.26).

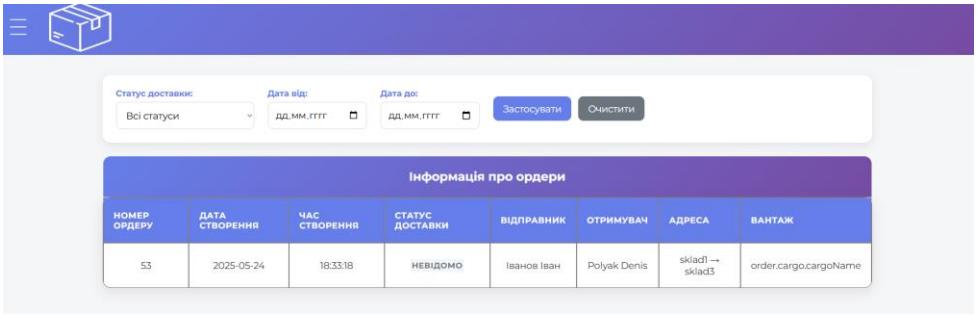


Рисунок Г.26 – Головна сторінка “Отримати вантаж”

Пошук потрібного замовлення: для швидкого пошуку потрібного замовлення використовуйте фільтр за статусом, розташований над таблицею. Виберіть відповідний статус зі списку: "До завантаження", "Завантажено", "До розвантаження", "Розвантажено" або "Завершено" або фільтр для визначення діапазону дат. Щоб побачити всі замовлення знову, виберіть опцію "Всі" у фільтрі (рисунок Г.27).

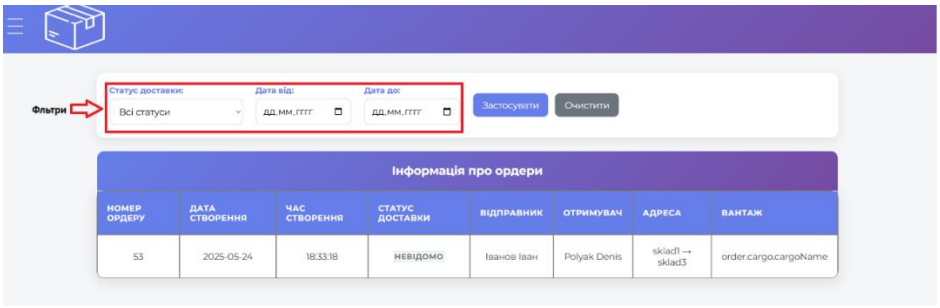


Рисунок Г.27 – Фільтри на головній сторінці “Отримати вантаж”

Перегляд детальної інформації про замовлення: щоб переглянути повну інформацію про конкретне замовлення, двічі клацніть на відповідному рядку в таблиці. У цьому вікні ви знайдете повну інформацію про відправника. Також тут відображається аналогічна інформація про отримувача, точні адреси відправлення та доставлення, а також детальні характеристики вантажу: назву, вагу, об'єм та кількість (рисунок Г.28).

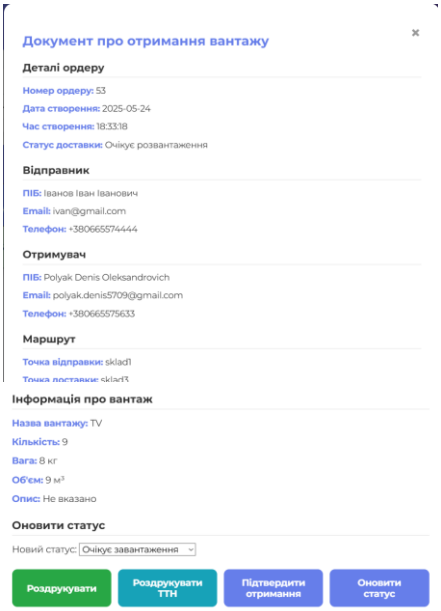


Рисунок Г.28 – Інформація про замовлення сторінка “Отримати замовлення”

Оновлення статусу замовлення: якщо вам потрібно оновити статус замовлення, знайдіть розділ "Оновити статус" у нижній частині детального вікна. Виберіть новий статус зі спадного списку, який містить всі доступні опції від "Чернетка" до "Завершено". Ви отримаєте повідомлення про успішне оновлення або про помилку, якщо щось пішло не так (рисунок Г.29).

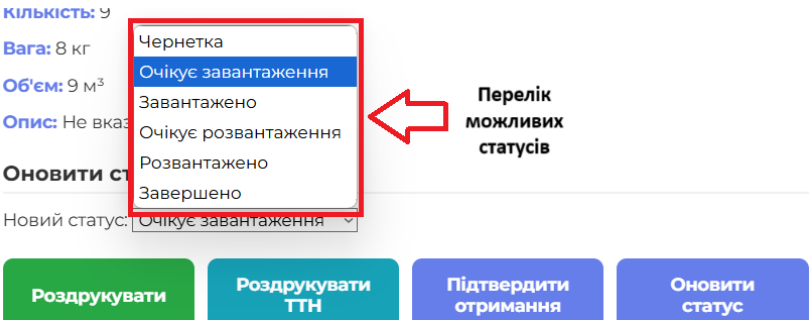


Рисунок Г.29 – Перелік можливих статусів заявки

Підтвердження отримання вантажу: коли ви фактично отримали вантаж, обов'язково підтвердіть це в системі. Для цього виберіть відповідне замовлення у таблиці (воно має бути виділене кольором) та натисніть кнопку "Підтвердити отримання" у вікні з деталями замовлення. Вебзастосунок зареєструє факт отримання вантажу та автоматично оновить статус замовлення.

Створення та друк документів: вебзастосунок дозволяє створювати офіційні документи про отримання вантажу. Після підтвердження отримання ви можете надрукувати документ, натиснувши кнопку "Роздрукувати" у детальному вікні. Це відкриє стандартне вікно друку вашого браузера, де ви зможете вибрати принтер та налаштування друку. При натисканні на кнопку “Роздрукувати” буде виведено наступне (рисунок Г.30).

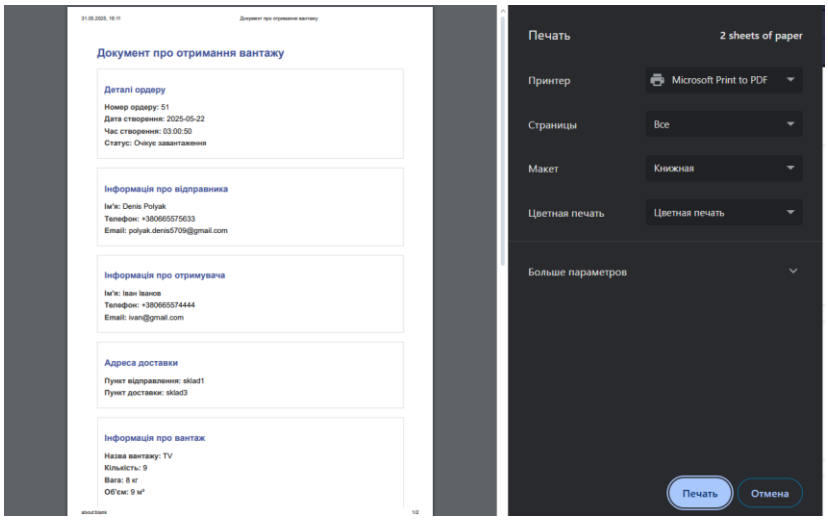


Рисунок Г.30 – Документ про отримання замовлення готовий до друку

При натисканні на кнопку “Роздрукувати ТТН” буде виведено наступне (рисунок Г.31).

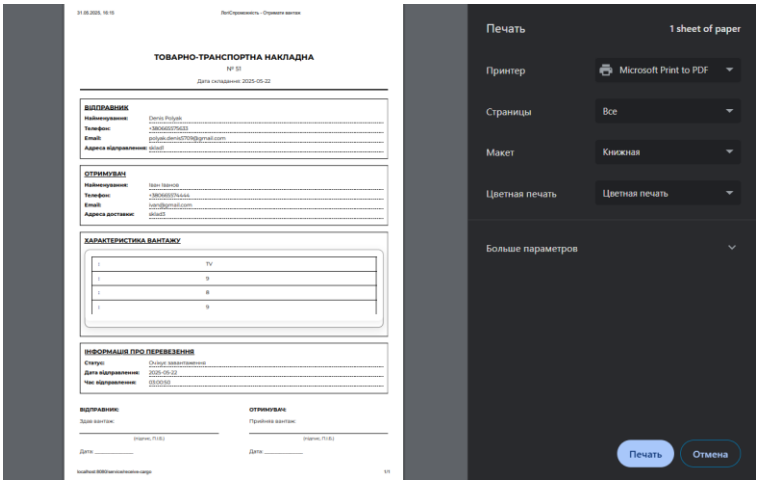


Рисунок Г.31 – Документ “Товарна транспортна накладна” готовий до друку

Детальний опис перегляду статусу вантажу

Сторінка “Статус вантажу” призначена для зручного відстеження статусів замовлень, що дозволяє ефективно управляти своїми вантажами. Основні функції включають можливість фільтрації замовлень за різними критеріями, такими як статус (чернетка, очікує завантаження, завантажено, очікує розвантаження, розвантажено, завершено) та дати (рисунок Г.32).

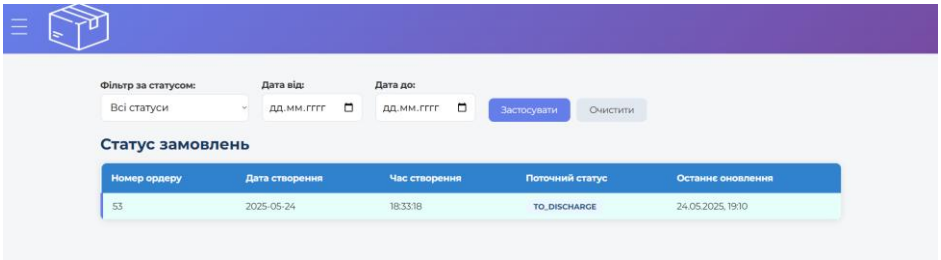


Рисунок Г.32 – Головна сторінка “Статус замовлення”

Користувач може вибрати статус замовлення з розкривного списку "Фільтр за статусом", а також вказати дати у полях "Дата від" та "Дата до" для обмеження результатів за часом. Після вибору необхідних параметрів, користувач натискає

кнопку "Застосувати", щоб активувати фільтри. Якщо потрібно скинути всі фільтри, можна натиснути кнопку "Очистити" (рисунок Г.33).

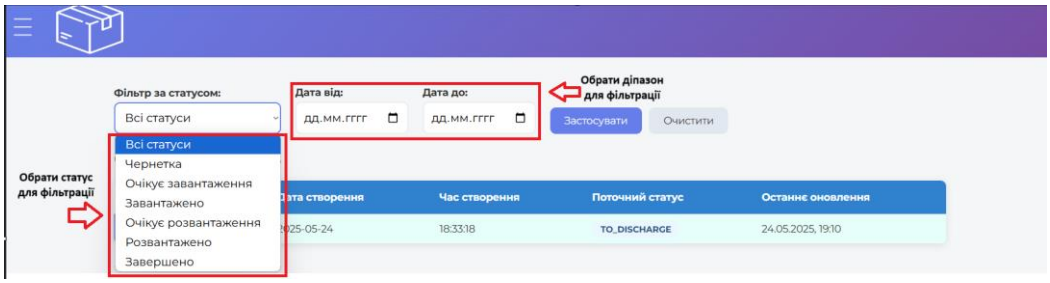


Рисунок Г.33 – Фільтри для сортування, сторінка “Статус замовлення”

Вебзастосунок надає можливість перегляду деталей замовлення. Для цього користувачеві потрібно знайти потрібне замовлення у таблиці та двічі натиснути на рядок з цим замовленням. Відкриється модальне вікно, в якому відображатимуться деталі замовлення, включаючи номер, дату, час, поточний статус та історію статусів (рисунок Г.34).

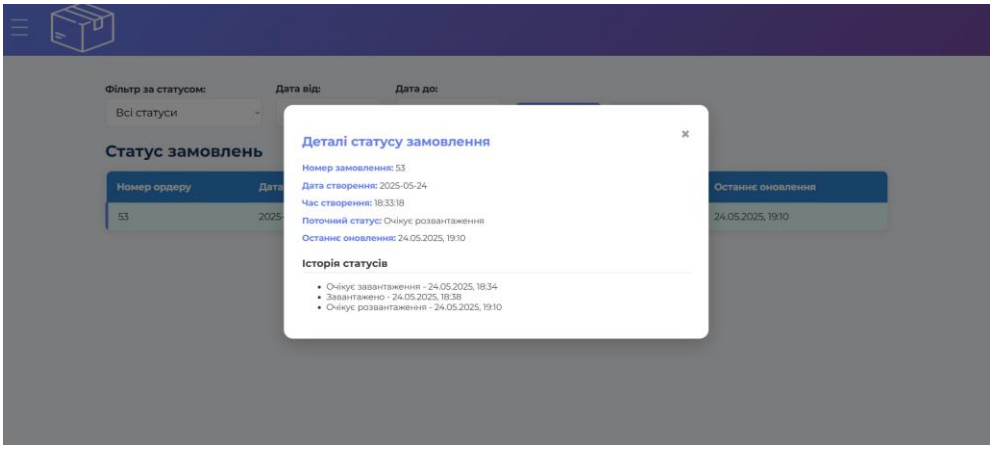


Рисунок Г.34 – Модальне вікно з деталями та історією змін статусі

Детальний опис перегляду історії замовлень

Сторінка "Історія замовлень" призначена для перегляду замовлень, які вже завершені або були відхилені. На цій сторінці відображаються лише ті заявки, які пройшли повний цикл обробки й мають фінальний статус – «Виконано» або «Відхилено» (рисунок Г.35).

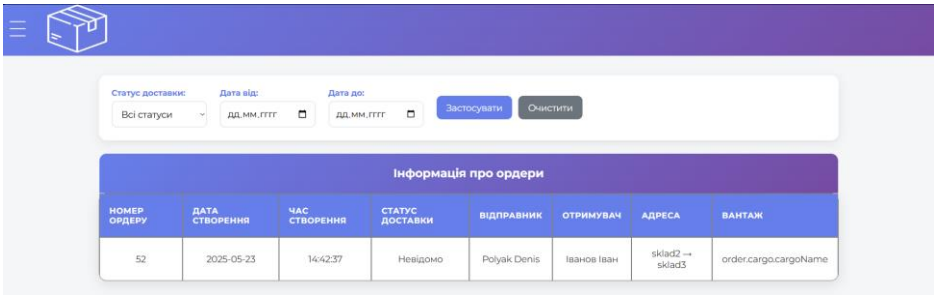


Рисунок Г.35 – Головна сторінка “ Історія замовлень ”

У таблиці на екрані представлено основну інформацію про кожне замовлення: номер заявки, дата її створення, фінальний статус, а також дані про вантажовідправника та вантажоотримувача.

При подвійному кліку на запис в таблиці відкриється документ із детальною інформацією про замовлення (рисунок Г.36), його можна роздрукувати в поточній формі і у формі ТТН. Кнопки “Роздрукувати” та “Роздрукувати ТТН” працює аналогічно як і на сторінці “Отримати вантаж”.

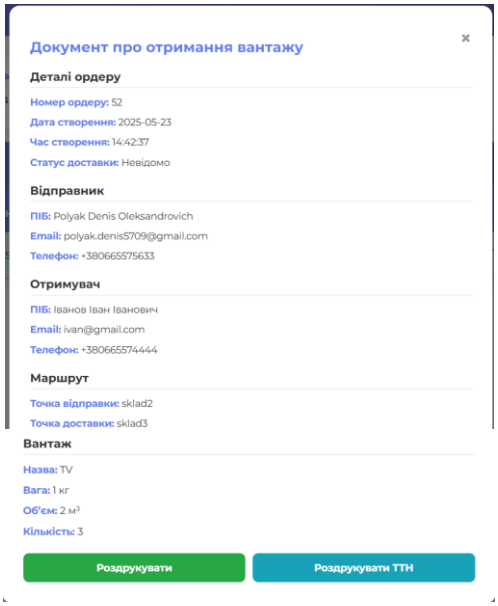


Рисунок Г.36 – Детальна інформація про замовлення сторінка “Історія замовлень”

Користувач також може скористатися можливостями фільтрації та сортування для швидкого пошуку потрібної інформації, слід зауважити що

порівняно зі сторінками “Отримати замовлення” та “Статус замовлення” тут лише 2 статуси тому що сторінка призначена для відображення завершених або відхилених замовлень (рисунок Г.37).

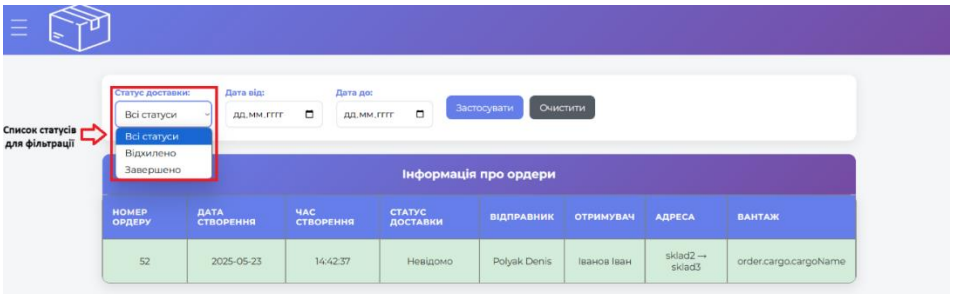


Рисунок Г.37 – Список статусів замовлень для фільтрації

Детальний опис перегляду аналітичних звітів

Сторінка "Аналітичні звіти" призначена для комплексного аналізу діяльності логістичної системи. Вона дозволяє генерувати детальні звіти, візуалізувати дані та експортувати результати в різних форматах. Для доступу до модуля перейдіть на сторінку аналітичних звітів через головне меню та переконайтеся, що у вас є необхідні права доступу (рисунок Г.38).

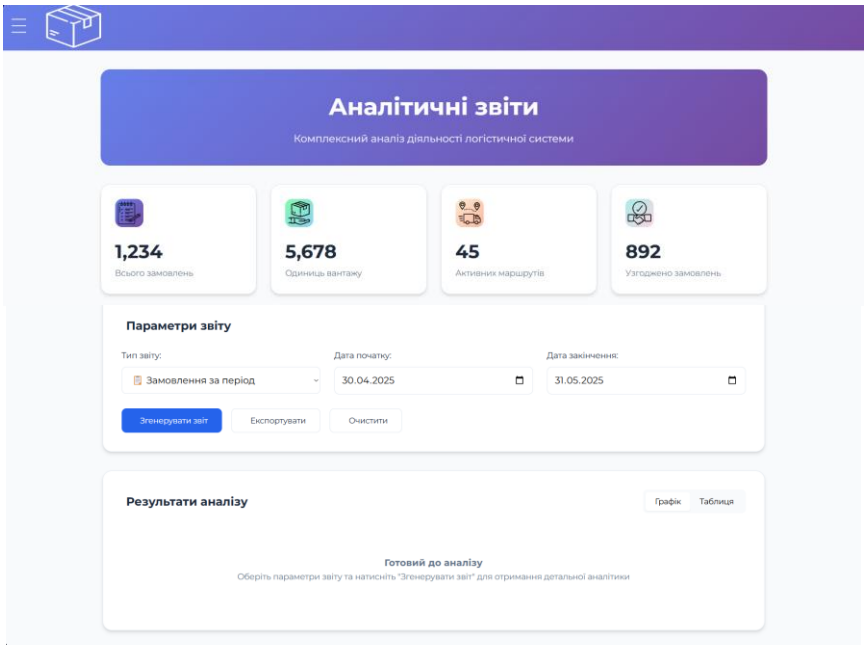


Рисунок Г.38 – Головна сторінка “Звіти”

У верхній частині сторінки відображаються чотири ключові показники у вигляді статистичних карток. Перша картка "Всього замовлень" показує загальну кількість замовлень у системі. Друга картка "Одиниць вантажу" відображає кількість оброблених вантажних одиниць. Третя картка "Активних маршрутів" показує поточну кількість активних логістичних маршрутів. Четверта картка "Узгоджено замовлень" відображає кількість успішно узгоджених замовлень. Ці картки автоматично оновлюються при завантаженні сторінки та відображають актуальні дані системи (рисунок Г.39).

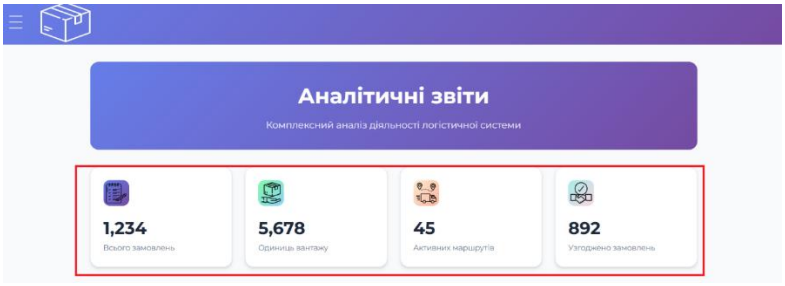


Рисунок Г.39 – Статистичні картки сторінка “Звіти”

Для генерації звіту спочатку оберіть тип звіту з розкривного списку. Доступні п'ять типів звітів: "Замовлення за період" для аналізу статусів замовлень, "Аналіз вантажів" для розподілу за типами вантажу, "Логістичні маршрути" для аналізу ефективності маршрутів, "Узгоджено замовлень" для перегляду активності користувачів та "Показники ефективності" для ключових метрик системи (рисунок Г.40).

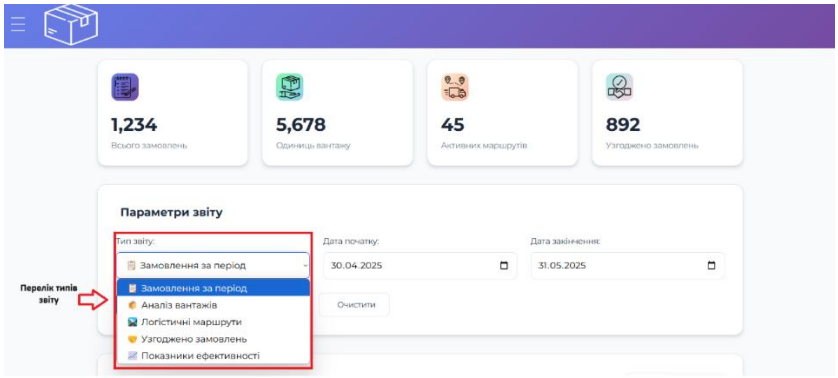


Рисунок Г.40 – Наявний перелік типів звіту сторінка “Звіти”

Після вибору типу звіту встановіть період аналізу, заповнивши поля "Дата початку" та "Дата закінчення". Для ефективної роботи рекомендується використовувати розумні періоди аналізу, не більше року за раз, щоб уникнути перевантаження системи (рисунок Г.41).

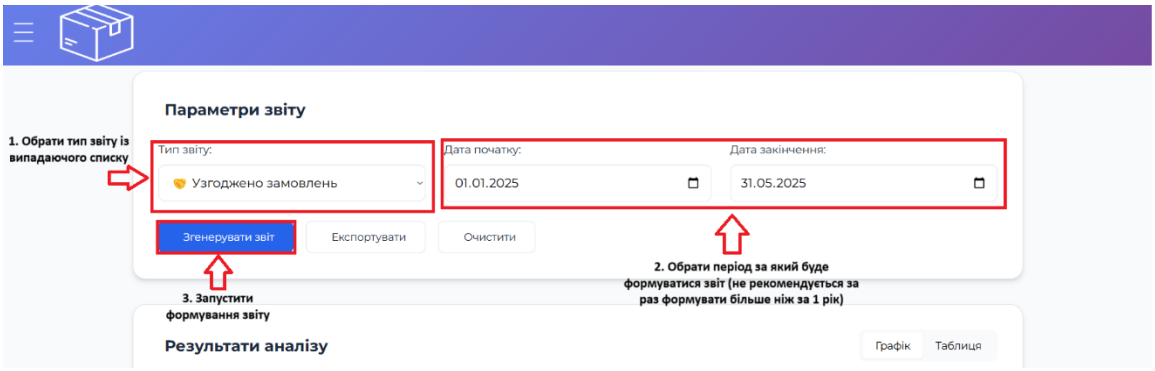


Рисунок Г.41 – Дії для запуску генерації звіту сторінка “Звіти”

Після заповнення всіх параметрів натисніть кнопку "Згенерувати звіт". Вебзастосунок покаже індикатор завантаження з текстом "Генерація звіту..." та після обробки даних відобразить результати в секції "Результати аналізу".

Результати звіту можна переглядати у двох режимах. Графічний режим відображає дані у вигляді інтерактивних діаграм - кругових для розподілу даних або стовпчастих для порівняння значень (рисунок Г.42).

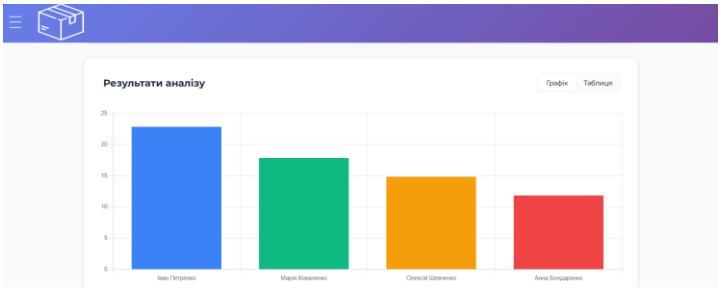


Рисунок Г.42 – Графічне представлення результатів аналізу сторінка “Звіти”

Табличний режим представляє дані у структурованому вигляді з можливістю детального аналізу. Таблиця містить заголовки стовпців відповідно до типу звіту та рядки з даними. При наведенні курсора на рядки вони підсвічуються для кращої

читабельності. Цей режим особливо корисний для точного аналізу числових значень (рисунок Г.43).

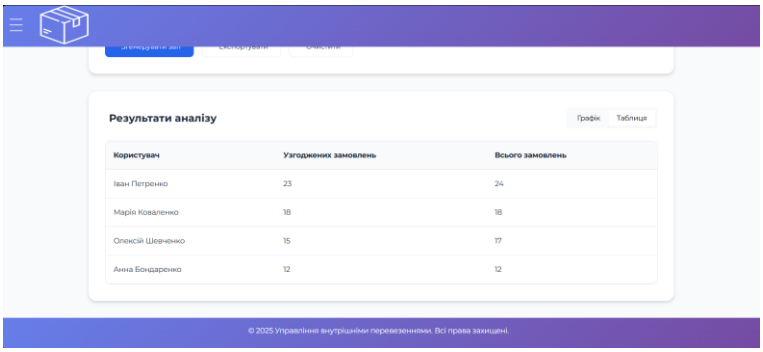


Рисунок Г.43 – Табличне представлення результатів аналізу сторінка “Звіти”

Переключення між режимами відображення здійснюється за допомогою кнопок у правому верхньому куті секції результатів. Кнопка "Графік" активує візуальний режим з діаграмами, а кнопка "Таблиця" переключає на табличний перегляд. Активний режим підсвічується, і дані автоматично форматуються згідно з обраними представлення при переключенні (рисунок Г.44).

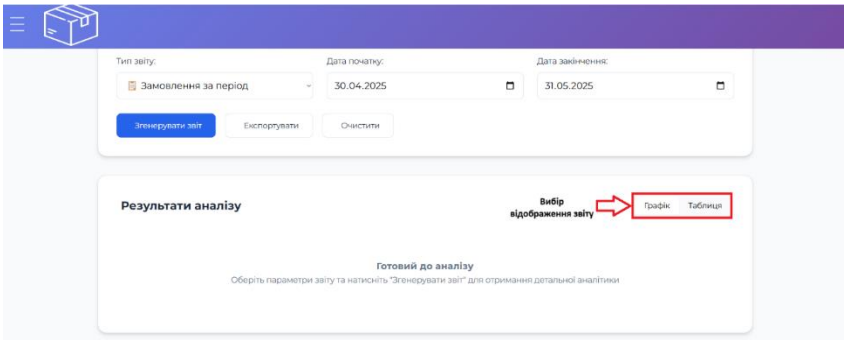


Рисунок Г.44 – Вибір представлення результатів аналізу сторінка “Звіти”

Функція експорту дозволяє зберегти згенеровані звіти в різних форматах. Для початку експорту натисніть кнопку "Експортувати", але спочатку переконайтеся, що звіт вже згенеровано. Якщо звіт не створено, вебзастосунок покаже повідомлення про помилку з проханням спочатку згенерувати звіт (рисунок Г.45).

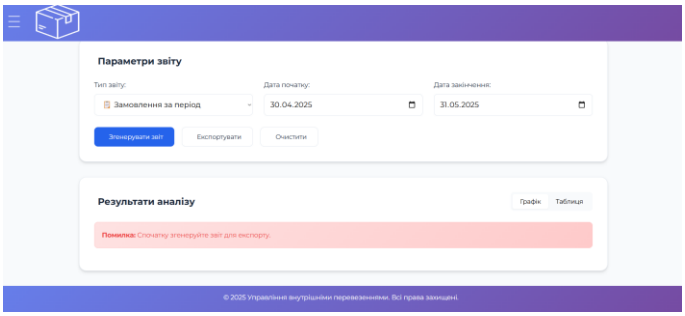


Рисунок Г.45 – Експортування результатів аналізу (звіту) без попередньої генерації сторінка “Звіти”

При натисканні кнопки експорту відкривається модальне вікно з шістьма варіантами форматів (рисунок Г.46).

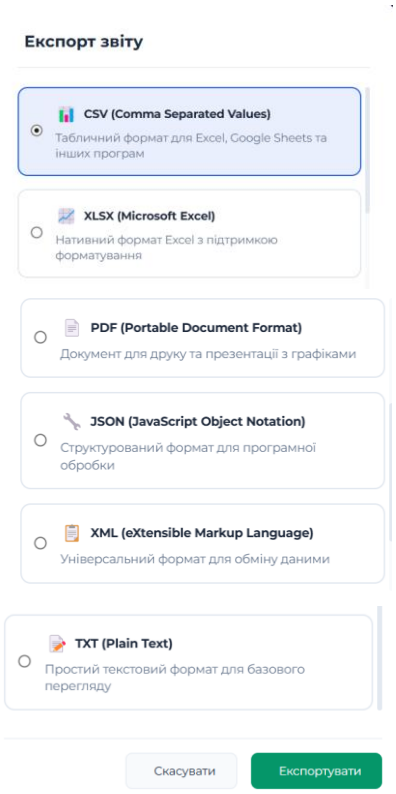


Рисунок Г.46 – Модальне вікно з варіантами форматів експорту звіту

Для вибору формату клікніть на відповідний варіант у модальному вікні. Обраний формат буде підсвічений синім кольором. Після вибору натисніть кнопку "Експортувати" для початку процесу. Вебзастосунок покаже індикатор завантаження з текстом "Експорт..." та після завершення автоматично завантажить файл на ваш пристрій (рисунок Г.47).

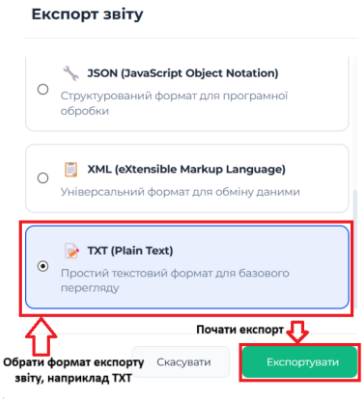


Рисунок Г.47 – Пояснення кроків в модальному вікні для експорту звіту

Функція очищення звіту доступна через кнопку "Очистити" та повертає інтерфейс до початкового стану. При очищенні видаляється поточний звіт, знищується діаграма, скидаються перемикачі режимів перегляду та відображається початкове повідомлення про готовність до аналізу. Ця функція корисна для початку роботи з новим звітом (рисунок Г.48).

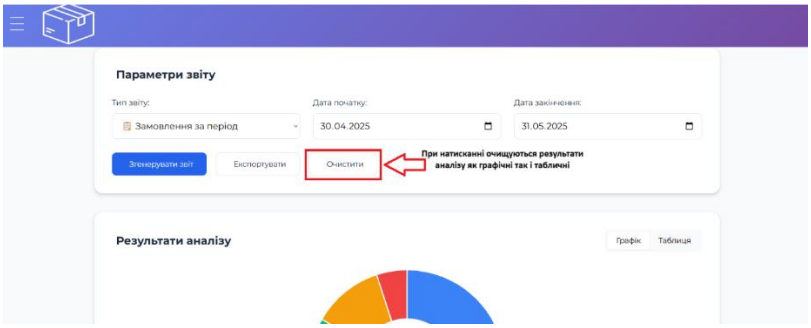


Рисунок Г.48 – Очистити звіт сторінка “ Звіти ”

При роботі з системою можуть виникати різні помилки. Помилка "Будь ласка, оберіть обидві дати" з'являється при незаповнених полях дат і вирішується заповненням обох полів. Помилка "Спочатку згенеруйте звіт для експорту" з'являється при спробі експорту без створеного звіту (рисунок Г.49).

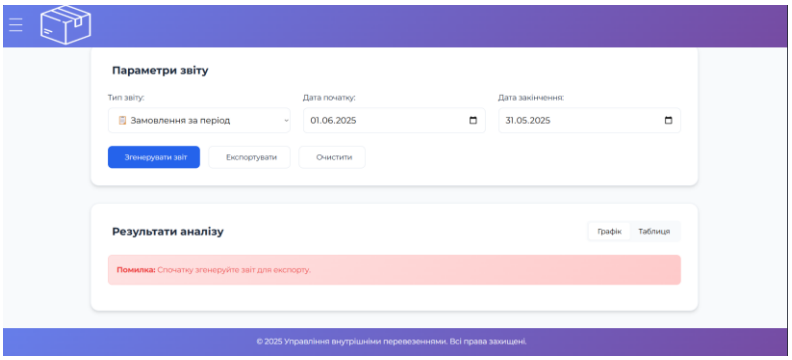


Рисунок Г.49 – Інформування користувача через неправильну послідовність дій під час експорту звіту сторінка “ Звіти ”

Для великих обсягів даних краще використовувати табличний режим, оскільки він швидше обробляється.

3.3 Завершення роботи програми

Для завершення роботи, по бажанню, можна вийти з особистого акаунту на сторінку авторизації вибравши в навігаційному меню кнопку “Вийти” (рисунок Г.50).

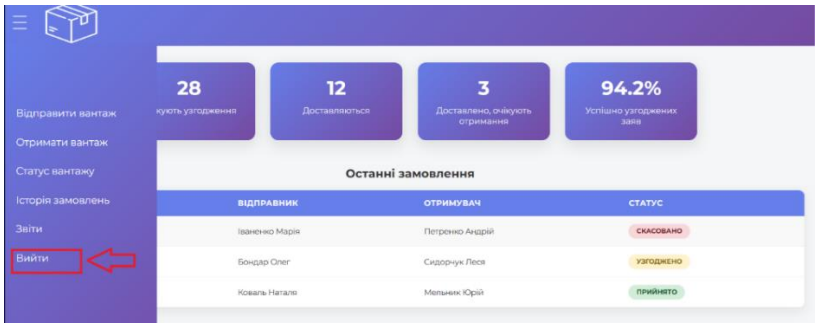


Рисунок Г.50 – Вихід з особистого акаунта користувача

Після натиснення на кнопку вийти буде здійснено завершення сесії та перехід на сторінку авторизації користувача (рисунок Г.1)

ДОДАТОК Д
ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ВЕБЗАСТОСУНКА ДЛЯ ВНУТРІШНІХ ПЕРЕВЕЗЕНЬ КОМПАНІЇ «AGROFUSION»

1 Об'єкт випробувань

1.1 Найменування випробуваної програми

Об’єктом випробувань є програмне забезпечення під найменуванням: вебзастосунок для внутрішніх логістичних перевезень компанії “Agrofusion”.

1.2 Область застосування випробуваної програми

Вебзастосунок призначений для автоматизації процесів управління внутрішніми логістичними перевезеннями в межах компанії “Agrofusion”.

Програма використовується логістичними відділами різних підрозділів підприємства для:

– оформлення та обробки заявок на перевезення;

– контролю руху вантажів між підрозділами;

– ведення електронної документації (ТТН);

– забезпечення погодження заявок між відповідальними особами.

Застосовується у внутрішньому корпоративному середовищі з доступом через локальну мережу підприємства.

1.3 Позначення випробуваної програми.

Позначення програми: AF-LOG-INT-2025

2 Мета випробувань вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion”.

Метою проведення випробувань є перевірка працездатності, функціональної повноти та відповідності вебзастосунка для внутрішніх перевезень компанії “Agrofusion” згідно з вимогами технічного завдання. Випробування здійснюються для оцінки коректності виконання основних функцій системи, зокрема створення та обробки замовлень, зміни їх статусів, погодження перевезень, генерації

супровідних документів, а також перевірки взаємодії клієнтської та серверної частин.

Додатково перевіряється зручність користування, стабільність роботи, цілісність даних при взаємодії з базою даних та відповідність інтерфейсу вимогам кінцевих користувачів. Результати випробувань дозволяють зробити висновок про готовність програмного забезпечення до використання у виробничому середовищі.

3 Вимоги до вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion”

У цьому розділі наведено перелік функціональних та експлуатаційних вимог, які підлягають перевірці під час випробувань програмного забезпечення згідно з технічним завданням.

Перелік функцій для перевірки:

– Створення/редагування заявки на перевезення з усіма обов’язковими параметрами.

– Збереження інформації про статус заявки та її зміну.

– Узгодження заявки з відповідальними особами.

– Ведення історії змін та фіксація подій (логування).

– Фільтрація та пошук за параметрами.

Перелік функціональних вимог згідно з технічним завданням (ТЗ):

– Створення заявок на вантажоперевезення.

– Введення параметрів заявки: тип вантажу, обсяг, пункти відправлення та призначення, бажаний час доставлення, спеціальні умови.

– Погодження заявок відповідальною особою;

– Відстеження статусу заявок;

– Формування стандартних звітів про перевезення;

– Експорт звітів;

4 Вимоги до програмної документації

Склад програмної документації, що повинна супроводжувати програмне забезпечення:

Технічне завдання – визначає мету, функціональні та технічні вимоги до програмного забезпечення, а також умови його розробки та впровадження.

Програма та методика випробувань – містить план і методи тестування, а також критерії оцінки відповідності програми вимогам технічного завдання.

Керівництво користувача – забезпечує інструкції для користувачів щодо встановлення, налаштування та роботи з вебзастосунком.

Опис програмного забезпечення – дає технічний опис структури, алгоритмів, функціональних модулів і логіки роботи програми.

Вихідні тексти програмних модулів – містить програмний код, що реалізує функціональність застосунку.

Спеціальні вимоги до програмної документації відсутні в ТЗ.

5 Засоби і порядок випробувань

Технічні засоби:

Випробування проводяться на робочих станціях із операційними системами Windows 10 або вище, macOS Big Sur і новіших, або Linux Ubuntu 20.04 та вище. Мінімальні технічні характеристики комп’ютерів: процесор Intel Core i3 або еквівалент, 4 ГБ оперативної пам’яті (рекомендовано 8 ГБ), 2 ГБ вільного місця на жорсткому диску. Для перевірки клієнтської частини застосовується сучасні веббраузери: Google Chrome, Mozilla Firefox, Microsoft Edge. Серверна частина розміщена на локальному сервері з операційною системою Linux (наприклад, Ubuntu Server) та СУБД MySQL.

Програмні засоби: IntelliJ IDEA, JUnit 5, JMeter MySQL Workbench, сучасні веббраузери (Google Chrome, Firefox, Edge).

Порядок проведення випробувань.

Підготовчий етап:

– Встановлення та налаштування тестового середовища (сервер, база даних, клієнтські машини).

– Підготовка тестових даних у базі (користувачі з різними ролями, заявки, документи).

– Перевірка працездатності середовища.

Функціональне тестування:

– Виконання модульних тестів (JUnit) для перевірки бізнес-логіки серверної частини (створення, редагування заявок, генерація документів, перевірка прав доступу).

– Тестування REST API через Postman для перевірки коректності обробки HTTP-запитів (GET, POST).

– Перевірка роботи клієнтської частини в браузері: створення, погодження, скасування заявок, формування звітів.

Інтеграційне тестування:

– Перевірка взаємодії клієнтської і серверної частин, коректність обробки даних між ними.

Навантажувальне тестування:

– Запуск тестів у JMeter для імітації одночасної роботи великої кількості користувачів.

– Аналіз часу відповіді сервера, стабільності роботи під навантаженням, визначення межі продуктивності.

Безпекове тестування:

– Перевірка механізмів автентифікації та авторизації, захист від несанкціонованого доступу.

Підсумковий аналіз:

– Збір і документування результатів тестування.

– Виявлення та фіксація дефектів.

– Корекція виявлених помилок та повторне тестування.

6 Методи випробувань

Для перевірки відповідності вебзастосунка для внутрішніх логістичних перевезень компанії “Agrofusion” вимогам, визначеним у технічному завданні, мають бути застосовані такі методи випробувань:

Модульне тестування (Unit Testing)

Мета: Перевірка коректності роботи окремих функціональних блоків серверної частини.

Метод: Використання фреймворку JUnit 5 для створення та запуску автоматизованих тестів.

Опис: Юніт-тести охоплюють логіку створення заявок, зміну статусів, обробку невалідних даних, збереження історії.

В таблиці Д.1 представлено тест-кейси для реалізації автоматизованих тестів за допомогою JUnit 5.

Таблиця Д.1 – Кейси для автоматизованих тестів

ID тесту	**Назва тесту**	**Вхідні дані**	**Початкові умови**	**Очікуваний результат**	**Покриття гілки**
TC-WB-001	Зміна статусу на новий, валідний	orderId = 1, newStatusString = "TO_FEED"	Order зі статусом DRAFT	Повертає true, статус оновлено, історія збережена	order.getStatus() != newStatus → true
TC-WB-002	Замовлення не знайдено	orderId = 9999, newStatusString = "DONE"	Замовлення з таким ID не існує	Повертає false	orderOpt.isEmpty() → true
TC-WB-003	Введено невалідний статус (помилка перетворення enum)	orderId = 1, newStatusString = "INVALID"	Order існує	Повертає false	OrderStatus.valueOf(...) кидає IllegalArgumentException
TC-WB-004	Введено статус, який уже встановлено	orderId = 1, newStatusString = "DRAFT"	Order уже має статус DRAFT	Повертає true, змін не відбулось	order.getStatus() != newStatus → false

Тестування продуктивності (Performance Testing)

Мета: Перевірка стійкості системи під навантаженням, час відгуку.

Метод: Тестування за допомогою Apache JMeter.

Опис: Визначити, як вебзастосунок поводиться при одночасному зверненні великої кількості користувачів до REST API та вебінтерфейсу.

У таблиці Д.2 представлено тест-кейси, які мають бути використані під час випробувань.

Таблиця Д.2 – Кейси навантажувального тестування

ID тесту	**Назва тесту**	**Мета**	**Методика**	**Очікуваний результат**
TC-PT-001	Тест навантаження: 100 користувачів отримують список заявок	Перевірити стабільність при одночасних запитах до /api/user/{id}/orders	Імітація 100 паралельних запитів GET через JMeter	Середній час відповіді < 2 секунд, без помилок HTTP 5xx

Функціональне тестування (Functional Testing)

Мета: Перевірка відповідності поведінки системи вимогам технічного завдання.

Метод: Ручне тестування інтерфейсу користувача.

У таблиці Д.3 представлено тест-кейси, які мають бути використані під час випробувань.

Таблиця Д.3 – Black-box тестування

ID тесту	**Назва**	**Мета**	**Кроки**	**Очікуваний результат**
TC-BB-001	Відправлення заявки без вибору отримувача	Перевірити валідацію, якщо користувач не вибрав отримувача	1. Увійти в систему. 2. Перейти на сторінку «Відправити вантаж» Заповнити форму, але не вказати отримувача 4. Натиснути “Надіслати заявку”	Виводиться повідомлення: Оберіть один із пунктів списку
TC-BB-002	Реєстрація email без "@"	Перевірити, що не можна зареєструватися з некоректною email-адресою	1. Перейти на сторінку реєстрації 2. Заповнити всі поля, але email = testemail.com3. Натиснути “Зареєструвати”	З’являється повідомлення про помилку формату email. Реєстрація не відбувається.
TC-BB-003	Створення заявки без назви вантажу	Перевірити, що без назви вантажу заявка не створиться	1. Перейти на /send-cargo2. Заповнити всі поля, крім поля “Назва вантажу”3. Натиснути “Надіслати заявку”	Виводиться повідомлення: Заповніть це поле над полем назви вантажу