НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
Інфомаційних систем і технологій

_____ Швиденко М.З.,,к.е.н.,доц.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка мобільного додатку інформаційної системи управління логістичною компанією

Спеціальність 122 «комп'ютерні науки»

Гарант освітньої програми

к.т.н. доц. _____\ Руденський Р.А
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

д-р філос., доц. _____\ Корольчук В.І
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Шевченко С. В
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри інформаційних
систем і технологій

_____/ Швиденко М.З.,,к.е.н.,доц.

підпис

“ _ ” _____2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студент Шевченко Сергій Володмирович

Спеціальність 122 «Комп’ютерній науки»

Тема бакалаврської кваліфікаційної роботи: Розробка мобільного додатку інформаціїйної системи управління логістичною компанією

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2246 “С”

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до роботи: опис програмного забезпечення

Перелік питань , які потрібно розробити:

Аналіз проблемної області, вибір та обґрунтування засобів для розробки системи, проектування інформаційної системи.

Дата видачі завдання “16” 12 2024р.

Керівник бакалаврської кваліфікаційної роботи

д-р філос., доц _____ Корольчук В.І
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Шевченко С.В
(підпис) (ПІБ)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6
ВСТУП 8
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 11
1.1 Опис предметної області 11
1.2 Огляд інформаційних джерел та існуючих рішень 13
1.3 Аналіз вимог до програмної системи 18
1.4 Постановка завдання 21
1.5 Моделювання предметної області 22
2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 28
2.1 Логічна модель даних у вигляді ER-діаграми 28
2.2 Діаграма класів і кооперації 31
2.3 Діаграма пакетів 35
2.4 Діаграма компонентів 37
3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 41
3.1 Система управління базою даних 41
3.2 Розробка інформаційної бази 42
3.3 Архітектура програмного забезпечення 45
3.4 Вибір інструментарію для створення прикладного програмного забезпечення 47
3.5 Алгоритмізація програмних модулів 48
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ ПРОГРАМНОЇ СИСТЕМИ 52
4.1 Тестування системи 52
4.2 Вимоги до апаратного та програмного забезпечення 57
ВИСНОВКИ 61

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ 63
ДОДАТОК А 66

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

- API – Application Programming Interface – інтерфейс прикладного програмування
- ERP – Enterprise Resource Planning – система планування ресурсів підприємства
- FI – Financial Accounting – фінансовий облік (модуль ERP-системи SAP)
- GIS – Geographic Information System – геоінформаційна система
- GPS – Global Positioning System – глобальна система позиціонування
- MM – Materials Management – управління матеріальними ресурсами (модуль SAP)
- MVC – Model–View–Controller – архітектурний шаблон «Модель–Подання–Контролер»
- SAP TM – SAP Transportation Management – модуль SAP для управління перевезеннями
- SaaS – Software as a Service – програмне забезпечення як послуга
- SD – Sales and Distribution – управління продажами і дистрибуцією (модуль
- SAP)
- SQL – Structured Query Language – мова структурованих запитів до баз даних
- Swing – Стандартна графічна бібліотека Java для побудови інтерфейсів користувача
- СУБД – Система управління базами даних
- ТЗ – Технічне завдання
- UI/UX – User Interface / User Experience – інтерфейс користувача та досвід його використання

- UML – Unified Modeling Language – уніфікована мова моделювання

ВСТУП

У контексті цифрової трансформації бізнес-процесів та підвищення вимог до оперативності обробки логістичних операцій особливого значення набуває автоматизація управління логістичними компаніями. Логістична діяльність охоплює багатокомпонентні процеси планування маршрутів, обробки замовлень, керування транспортними засобами, обліку складських залишків та забезпечення документообігу, що вимагає впровадження інформаційних систем з високим рівнем адаптивності, масштабованості та інтеграції з зовнішніми сервісами [5], [6], [18].

Актуальність розробки інформаційної системи управління логістичною компанією зумовлена потребою у створенні спеціалізованого інструментарію, який дозволяє централізовано керувати логістичними об'єктами в режимі реального часу, оптимізувати витрати на перевезення, покращити комунікацію між структурними підрозділами компанії та забезпечити прозорість усіх етапів ланцюга постачання [7], [14]. Існуючі рішення на ринку здебільшого є комерційними, складними у налаштуванні або не враховують специфіку локального ринку, що обґрунтовує необхідність створення адаптованого вітчизняного програмного забезпечення [6].

У проєкті **реалізується підхід** до інженерії програмного забезпечення на основі об’єктно-орієнтованого аналізу, моделювання та реалізації компонентів системи з використанням інструментів UML (Unified Modeling Language) [12], [22]. Це дозволяє формалізувати вимоги, структуру та поведінку системи через стандартні діаграми класів, компонентів, випадків використання тощо, що підвищує зрозумілість, підтримуваність і ефективність процесу розробки [16], [17], [26].

Програмна реалізація системи виконується засобами Java із застосуванням бібліотеки Swing для побудови графічного інтерфейсу користувача, що забезпечує міжплатформенність, високу інтерактивність та зручність у

використанні [9], [23], [28]. В якості бази даних обрано реляційну модель з використанням SQL-технологій, що забезпечує надійне зберігання, фільтрацію та обробку логістичних даних (маршрути, замовлення, клієнти, транспортні одиниці) [15], [14].

Об’єктом дослідження виступає процес автоматизованого управління логістичними операціями в межах підприємств малого та середнього бізнесу.

Предметом дослідження є методи моделювання архітектури, алгоритмів взаємодії підсистем, механізмів обміну даними та реалізації бізнес-логіки інформаційної системи управління логістичною компанією.

Метою кваліфікаційної роботи є розробка та програмна реалізація багатофункціональної інформаційної системи управління логістичною компанією з використанням засобів Java та UML-моделювання, що забезпечує інтегроване управління логістичними процесами, ведення обліку, взаємодію з клієнтами та оптимізацію ресурсів.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

1. Провести системний аналіз предметної області логістичних операцій і виявити основні функціональні вимоги до майбутньої системи.
2. Побудувати UML-діаграми для формалізації структури та функціонування системи, включаючи діаграми класів, прецедентів, компонентів і послідовностей [12].
3. Спроєктувати архітектуру програмного забезпечення, зокрема визначити підсистеми обробки маршрутів, керування замовленнями та моніторингу транспортних засобів.
4. Розробити графічний інтерфейс користувача з використанням Java Swing для забезпечення зручної взаємодії з системою [9], [23].
5. Реалізувати реляційну базу даних для зберігання логістичних даних та забезпечити підтримку запитів SQL.
6. Провести тестування системи, оцінити її працездатність, ефективність і відповідність вимогам функціональності.

7. Надати рекомендації щодо впровадження інформаційної системи на підприємстві, включаючи аспекти адаптації, навчання персоналу та подальшого масштабування.

Наукова новизна роботи полягає у поєднанні класичних логістичних принципів з сучасними методами програмної інженерії на базі об’єктно-орієнтованого моделювання, що дозволяє реалізувати модульну, гнучку та масштабовану інформаційну систему управління логістичною компанією. Практичне значення полягає у можливості адаптації створеного рішення до потреб конкретного підприємства з мінімальними витратами на впровадження та супровід.

Структура кваліфікаційної роботи складається з трьох основних розділів і висновків. У першому розділі висвітлено теоретичні основи логістичного програмного забезпечення, охарактеризовано принципи побудови логістичних систем, інструменти проєктування та стандарти UML-моделювання, що використовуються при формуванні інформаційної структури. У другому розділі подано проєктну модель системи, наведено результати побудови UML-діаграм, описано структуру бази даних, реалізовано логічну архітектуру, обґрунтовано вибір інструментів і підходів до реалізації. У третьому розділі розглянуто реалізацію прикладного програмного забезпечення, алгоритмізацію модулів, приклади програмного коду та процеси перевірки працездатності з урахуванням обраної архітектури. У висновках підсумовано основні результати роботи, оцінено досягнення поставленої мети, ефективність розробленої системи та можливості її подальшого розвитку.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Управління логістичними процесами є критично важливою складовою ефективного функціонування підприємств, що здійснюють діяльність у сфері транспортування, зберігання та доставки товарів. З розвитком цифрових технологій традиційні методи управління логістикою зазнають трансформації в бік автоматизації та цифровізації, що потребує впровадження інформаційних систем, здатних інтегрувати всі етапи логістичного ланцюга – від прийому замовлень до контролю за виконанням доставок і обліком складських залишків [5], [6].

Предметна область даного дослідження охоплює розробку десктопної інформаційної системи управління логістичною компанією, що працює в локальному середовищі без залежності від постійного інтернет-з’єднання. Такий підхід забезпечує підвищену автономність, гнучкість та конфіденційність даних, що є особливо важливим для підприємств малого і середнього бізнесу, які прагнуть до оптимізації витрат на впровадження складних та дороговартісних вебрішень [7], [14].

Програмний продукт реалізується із використанням мови програмування Java та графічної бібліотеки Swing, що дозволяє створити програму з повноцінним графічним інтерфейсом користувача [9], [23]. У межах системи планується реалізація таких ключових компонентів:

1. модуль обліку логістичних замовлень, що забезпечує збереження, редагування та пошук інформації про відправлення, вантажоодержувачів, маршрути і строки доставки;
2. Система управління транспортом, яка дозволяє реєструвати транспортні одиниці, фіксувати технічний стан, контроль техобслуговування та призначення водіїв;

3. Підсистема маршрутизації, що підтримує розрахунок оптимальних маршрутів та контроль виконання рейсів з можливістю подальшого аналізу ефективності [6], [20];

4. База даних, реалізована на основі вбудованої реляційної СУБД з підтримкою SQL-запитів, що забезпечує зберігання структурованої інформації про всі елементи логістичного процесу [14], [15];

5. Інтерфейс адміністратора, який надає доступ до ключових функцій налаштування, перегляду звітів, управління користувачами та ролями в системі.

Предметна область вимагає ефективної організації взаємодії між внутрішніми підсистемами через чітко визначену архітектуру, що базується на принципах об’єктно-орієнтованого підходу. Для моделювання структури програмного забезпечення застосовуються UML-діаграми, зокрема діаграми класів, випадків використання, компонентів та активності, які дозволяють формалізувати логіку функціонування та взаємозв’язки між елементами системи [16], [22].

Особливістю предметної області є необхідність врахування широкого спектра змінних параметрів, які впливають на логістичну діяльність – таких як маршрутна сітка, тип транспортного засобу, специфіка вантажу, погодні умови тощо. Це потребує реалізації гнучкої модульної архітектури системи з можливістю масштабування та адаптації під конкретні бізнес-потреби [18], [19].

У результаті реалізації програмного продукту очікується отримання засобу, який забезпечує централізоване управління логістичними процесами, автоматизацію рутинних операцій, покращення якості сервісу для клієнтів та зниження витрат на операційну діяльність компанії. Система може використовуватись у малих і середніх логістичних підприємствах, транспортних компаніях, складських комплексах і сервісних центрах доставки, де необхідна швидка обробка даних, оптимізація маршрутів і контроль транспортної інфраструктури.

1.2 Огляд інформаційних джерел та існуючих рішень

Сучасний етап розвитку транспортної логістики вимагає широкого застосування програмних інструментів для автоматизації управлінських процесів, підвищення ефективності маршрутного планування, оптимізації використання ресурсів та забезпечення контролю виконання замовлень у реальному часі. Інформаційні системи такого типу не лише спрощують обробку великого обсягу даних, а й сприяють стратегічному прийняттю рішень за рахунок аналітики, прогнозування та інтеграції з корпоративними платформами. У цьому підрозділі наведено огляд чотирьох інформаційних рішень, які виступають галузевими аналогами проєктованої системи, та проведено їх порівняльний аналіз із позицій функціональності, структури та призначення.

Одним із провідних рішень є SAP Transportation Management (SAP TM) – компонент корпоративного середовища SAP, який забезпечує наскрізне управління логістичними процесами на рівні підприємства. Система підтримує формування замовлень, розрахунок вантажопотоків, створення маршрутів, призначення водіїв і транспортних засобів, а також побудову графіків доставки. SAP TM інтегрується з внутрішніми бізнес-модулями (FI, MM, SD), що дозволяє реалізовувати повний цикл логістичних операцій. На рис. 1.1 представлено інтерфейс модуля, де відображається панель формування вантажних замовлень, дорожніх рейсів та інтерактивне 3D-завантаження транспортного засобу. Праворуч візуалізовано спосіб розміщення вантажу в кузові, що враховує об’єм, вагу та навантаження на осі. Така функція забезпечує зменшення простоїв транспорту та мінімізує ризики перевантаження [1], [2].

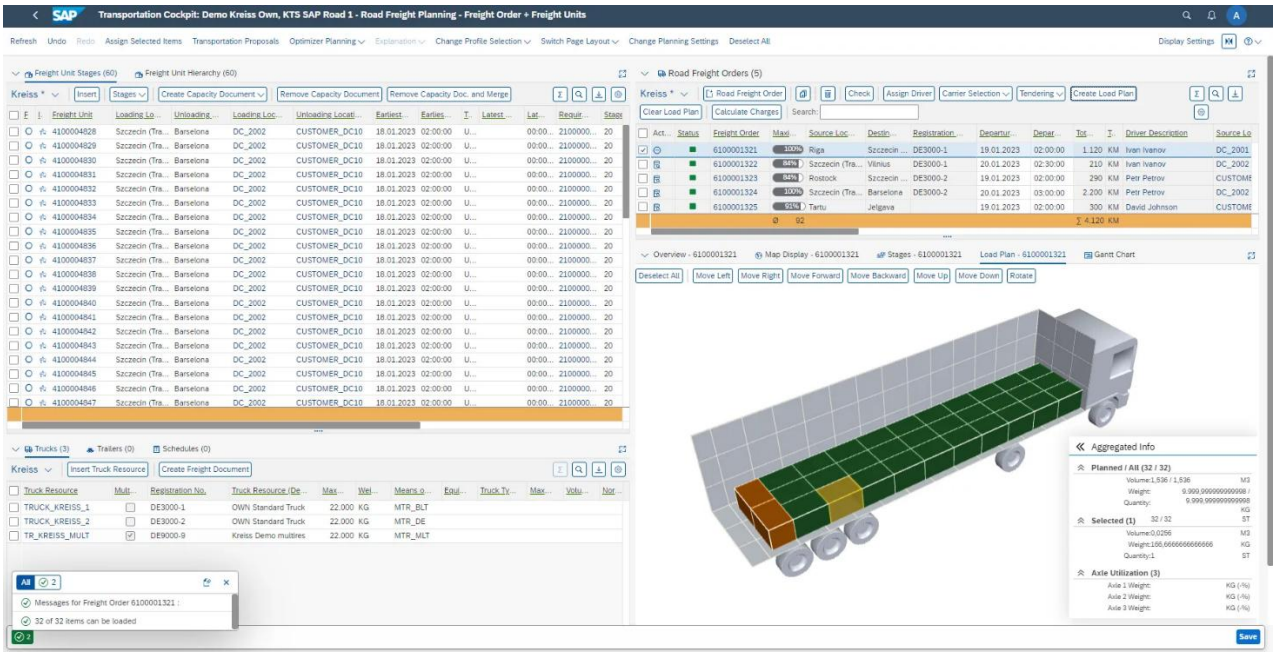

Рис 1.1 – Інтерфейс SAP TM: управління дорожніми перевезеннями та плануванням завантаження

Іншим потужним інструментом є Oracle Transportation Management, що є частиною Oracle Logistics Cloud. Ця система орієнтована на централізоване управління перевезеннями для середніх та великих компаній. Однією з ключових переваг є наявність модуля візуалізації маршрутів у геопросторовому середовищі з можливістю аналізу точок зупинок, часу прибуття, завантаженості доріг та витрат пального. Інтерфейс системи, поданий на рис. 1.2, ілюструє вікно управління доставками та маршрути водіїв у режимі реального часу. Кожен сегмент маршруту промарковано різними кольорами, що дозволяє швидко оцінювати ефективність логістичних операцій, контролювати простій та дотримання графіків [3].

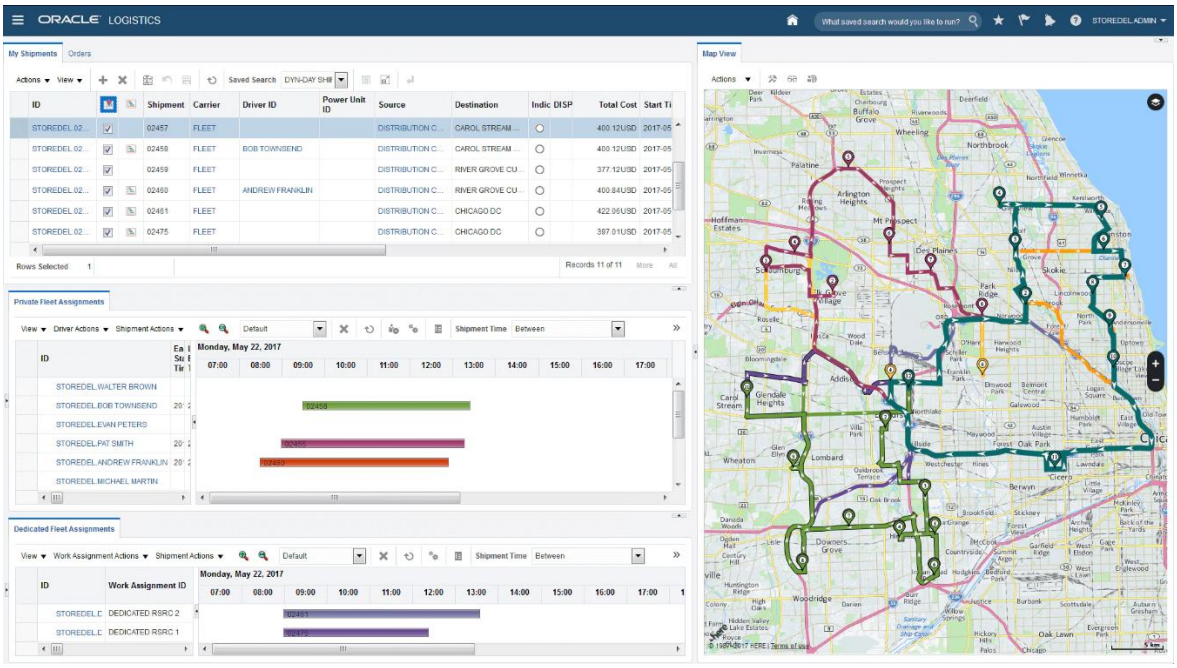

Рис. 1.2 – Інтерфейс Oracle Logistics Cloud: планування доставок з інтерактивною картою

Наступним прикладом є Transporeon – SaaS-платформа для логістичних провайдерів і вантажовідправників, що працює як транспортна біржа. Система забезпечує динамічне призначення вантажів, контроль транспорту, обмін повідомленнями, підтримку автоматичних ставок у тендерах та аналітичні модулі. На рис. 1.3 зображено декілька інтерфейсних панелей платформи, зокрема статистику активності замовників, графіки виконання замовлень та візуалізацію аналітичних показників. Такий підхід дозволяє оптимізувати бізнес-рішення завдяки інтерактивному представленню операційної інформації [4].

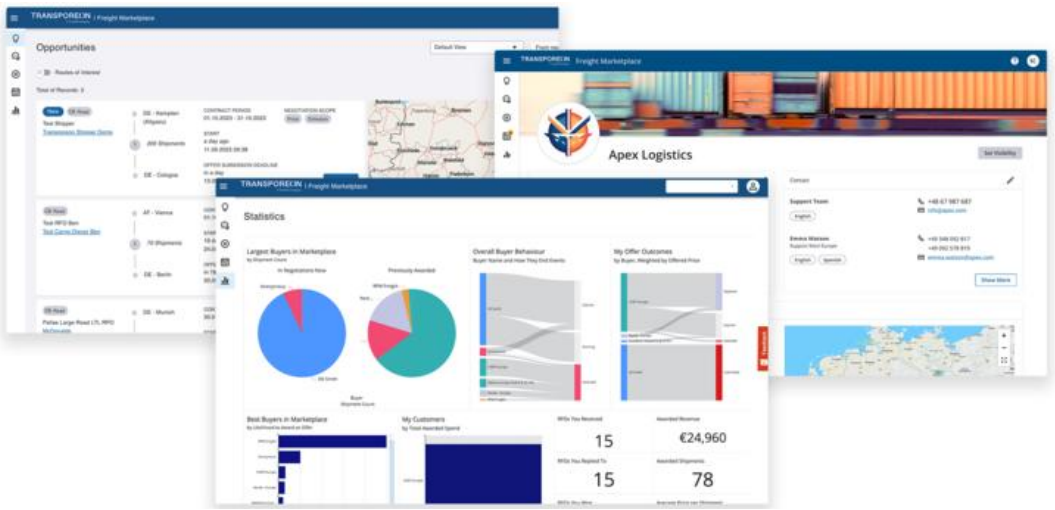

Рис. 1.3 – Панель керування Transporeon: маркетплейс вантажів і аналітика

Останнім прикладом є система WebFleet (раніше TomTom Telematics), яка орієнтована на управління автопарками та контролем за рухом ТЗ у реальному часі. WebFleet дозволяє відстежувати місцезнаходження транспорту, будувати оптимальні маршрути, аналізувати витрати пального та стиль водіння. Інтерфейс, представлений на рис. 1.4, демонструє карту маршруту з підрахунком витрати пального та сповіщеннями про відхилення. Ця інформація має вирішальне значення для диспетчерів, які відповідають за ефективність автопарку [5].

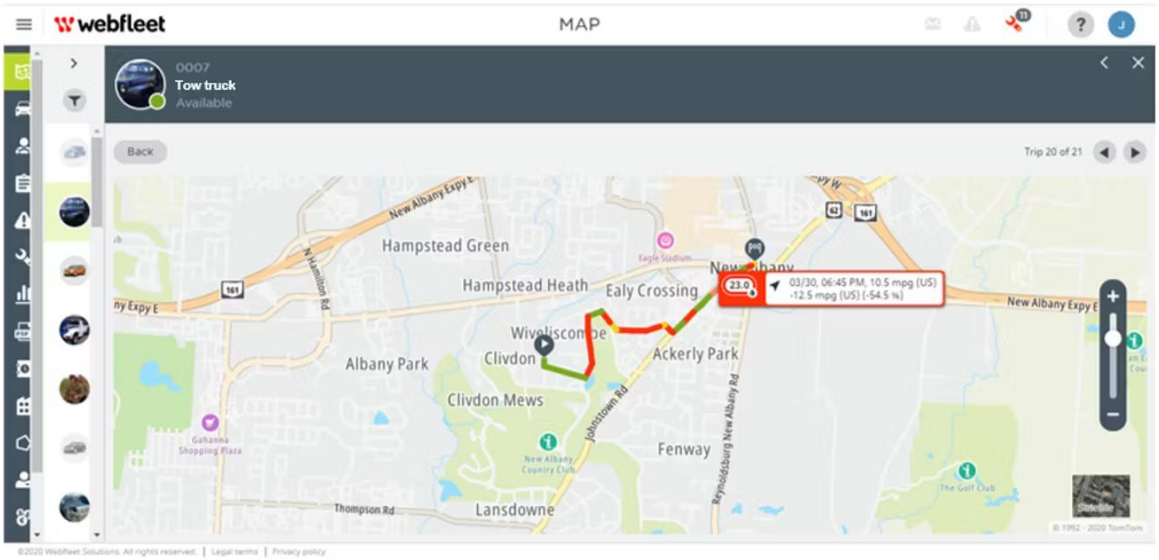

Рис. 1.4 – Інтерфейс WebFleet: контроль маршруту та аналітика витрат

З метою формалізації порівняння аналогів було здійснено аналіз чотирьох програмних рішень за низкою технічних та функціональних критеріїв: тип розгортання, підтримка візуалізації маршрутів, інтеграція з БД, автоматизація планування, контроль завантаження та наявність телематики. Порівняльна таблиця 1.1 представлена далі

Таблиця 1.1 - Порівняння програмних рішень

Критерій / Система	SAP TM	Oracle TMS	Transporeon	WebFleet
Тип рішення	ERP-модуль	Хмарна логістична платформа	SaaS-біржа перевезень	Telematics/Fleet Manager

Тип розгортання	Локальне/Хмарне	Хмарне	Хмарне	Хмарне

Продовження таблиці 1.1

Побудова маршрутів	✓	✓	✓	✓
Візуалізація маршрутів	Обмежено (3D-завантаження)	✓ (GIS + мапи)	✓	✓ (мапи, трекінг)
Автоматичне планування	✓	✓	Частково (на тендерах)	✕
Інтеграція з базою даних	SAP HANA, ERP	Oracle DB	API + зовнішні системи	Частково
Контроль завантаження ТЗ	✓ (3D-модель кузова)	✕	✕	✕
Вбудована телематика	✕	Частково (через API)	✕	✓
Цільова аудиторія	Виробничі корпорації	Великі логістичні компанії	Транспортні брокери	Флот-менеджери
Підтримка аналітики	✓ (BI-модулі)	✓	✓	✓

На основі проведеного аналізу можна зробити висновок, що жодна з розглянутих систем не охоплює повний спектр функціоналу, необхідного для невеликої або середньої логістичної компанії в умовах локальної автономної роботи без підключення до хмарних сервісів. Водночас кожне з рішень реалізує окремі функціональні модулі, які можуть бути адаптовані або враховані під час розробки власного програмного забезпечення.

Зокрема, доцільно врахувати наступне:

- від SAP TM – механізм формування та перевірки завантаження транспортного засобу із візуалізацією на основі 3D-моделі, а також багаторівнева структура транспортних замовлень.
- Від Oracle TMS – візуалізація маршрутів із застосуванням інтерактивної мапи та графіків навантаження ресурсів по часу.

- Від Transporeon – концепція прозорого управління тендерами та обміну даними між перевізниками й вантажовласниками, що може бути застосовано в якості комунікаційного модуля.
- Від WebFleet – модуль телематики та моніторингу маршрутів у реальному часі, який може бути використаний як орієнтир для побудови майбутньої інтеграції з GPS-модулями та трекерами.

Е, проєктована система буде орієнтована на поєднання кращих практик аналізованих рішень, проте адаптована для локального застосування, з акцентом на автономну роботу, просту інтеграцію, збереження маршрутних сценаріїв та гнучку візуалізацію логістичних процесів. Це дозволить створити програмне забезпечення, оптимізоване під потреби логістичних операторів, які не мають доступу до хмарних ERP-рішень або потребують кастомізованого підходу.

1.3 Аналіз вимог до програмної системи

Аналіз вимог до інформаційної системи управління логістичною компанією є ключовим етапом у процесі проєктування, оскільки саме на цьому етапі формалізуються очікування користувачів, функціональні можливості, технічні обмеження та вимоги до продуктивності. Для досягнення цілісного представлення вимог було використано стандартизований підхід до їх класифікації, зокрема рекомендації IEEE 830–1998 [1] та ISO/IEC/IEEE 29148:2018 [2], згідно з якими всі вимоги поділяються на функціональні, нефункціональні, системні та обмеження.

У рамках дослідження було сформовано основні функціональні вимоги, що описують поведінку системи при взаємодії з користувачами та іншими компонентами. Таблиця 1.2 містить перелік ключових функцій, реалізація яких є критичною для забезпечення повного циклу обробки логістичних замовлень.

Таблиця 1.2 – Функціональні вимоги до системи управління логістикою

№	Назва функції	Опис функціональності

1	Реєстрація замовлення	Оператор вводить дані про клієнта, маршрут, вимоги до ТЗ
2	Збереження у локальній базі	Дані зберігаються автономно в SQLite або іншій вбудованій СУБД

Продовження таблиці 1.2

3	Побудова оптимального маршруту	Система генерує маршрут на основі введених точок і дорожніх умов
4	Призначення водія та ТЗ	Вибір доступного ресурсу відповідно до характеристик маршруту
5	Генерація звітів	Створення звітів про виконані рейси, навантаження, простій
6	Вивід повідомлень	Інформування оператора про події (успішне збереження, помилки, перевантаження тощо)

Функціональні вимоги сформовані з урахуванням аналізу аналогів (див. підпункт 1.3) та очікувань типових користувачів – логістичних операторів, диспетчерів та адміністраторів [3].

Разом із тим, розробка якісної програмної системи передбачає визначення нефункціональних вимог, що регламентують обмеження та очікувані характеристики системи на рівні продуктивності, безпеки, зручності використання та підтримки. Вони подані в табл. 1.3.

Таблиця 1.3 – Нефункціональні вимоги до системи

№	Категорія	Вимога
1	Надійність	Система має відновлюватися після збою без втрати даних
2	Продуктивність	Час відповіді на запит користувача не повинен перевищувати 1 секунди
3	Безпека	Захист доступу до конфіденційної інформації через пароль та контроль ролей користувачів
4	Масштабованість	Підтримка розширення функціоналу без критичних змін у архітектурі
5	Зручність використання	Інтерфейс має бути інтуїтивно зрозумілим, з чіткою структурою та логікою навігації

Визначення цих вимог ґрунтується на стандартних рекомендаціях до проєктування UI/UX [4], а також досвіді впровадження логістичних систем у корпоративному середовищі [5].

Окрему увагу слід приділити системним вимогам, що визначають технічні характеристики середовища функціонування та технологічний стек, у якому буде реалізована система. Оскільки розробка відбувається у вигляді десктопної Java-програми з використанням Java Swing, вимоги до середовища є критичними для сумісності та продуктивності. Системні вимоги більш детально розглянуто далі у таблиці 1.4.

Таблиця 1.4 – Системні вимоги до середовища виконання

№	Компонент	Вимога
1	Операційна система	Windows 10/11, Linux Ubuntu 20.04+, macOS 11+
2	Віртуальна машина Java	Java SE 17 або новіша
3	Середовище зберігання даних	Вбудована СУБД (SQLite, H2), з можливістю подальшої інтеграції з PostgreSQL/MySQL
4	Графічна бібліотека	Java Swing + можливе підключення бібліотек GraphStream / JGraphX для візуалізації
5	Інтерфейс користувача	Локальний GUI з підтримкою багатомовності та тем оформлення

Врахування системних вимог дозволяє забезпечити стабільну роботу програми на більшості робочих станцій та підтримку ключових функцій без залучення сторонніх серверів чи хмарних платформ.

Крім наведеного, було визначено ряд обмежень, що випливають із технічного середовища та специфіки цільової аудиторії. Вони включають: автономний режим роботи без доступу до Інтернету; обмеження у використанні хмарних API; мінімізацію обсягу зовнішніх бібліотек; необхідність локалізації українською мовою. Ці аспекти було враховано при формуванні архітектури системи та доборі технологічного стеку.

1.4 Постановка завдання

На основі проведеного аналізу предметної області, моделювання основних сценаріїв взаємодії користувачів із системою та формалізованих вимог, сформульовано технічне завдання для розробки інформаційної системи управління логістичною компанією. Система має забезпечувати повний цикл обробки логістичних замовлень: від введення клієнтських і маршрутних даних до формування оптимального плану перевезення із відповідним візуальним та звітним супроводом. Вся обробка має виконуватися локально на робочому місці логістичного оператора без залучення зовнішніх сервісів чи підключення до мережі Інтернет.

Передбачається створення десктопного застосунку з графічним інтерфейсом на основі Java Swing, який взаємодіє з вбудованою СУБД (SQLite). У програмі реалізовуватимуться окремі програмні модулі: обробки замовлень, маршрутизації, управління транспортом, візуалізації даних та генерації звітності.

До вхідних даних, які система повинна приймати для обробки, належать клієнтські дані, параметри перевезення, характеристики вантажу та наявні ресурси. Ці параметри структуровано у таблиці 1.5.

Таблиця 1.5 – Вхідні дані для обробки в системі

Категорія	Параметри
Клієнт	ПІБ, телефон, ID клієнта
Замовлення	Місце відправлення, призначення, дата/час, примітки
Вантаж	Вага, об’єм, категорія вантажу (стандартний/спеціальний)
Транспортні ресурси	Перелік ТЗ із характеристиками (вантажопідйомність, об’єм, статус)
Водії	Ім’я, ID, доступність, водійська категорія
Дорожні умови	Задання користувачем або вибір із заздалегідь збережених шаблонів

Результатом роботи системи має бути створення повного об'єкта замовлення з маршрутною інформацією, призначенням ресурсу і можливістю візуального та текстового подання. Основні результати, які очікуються від функціонування системи, подано в таблиці 1.6.

Таблиця 1.6 – Очікувані результати роботи системи

Компонент результату	Опис
Об'єкт замовлення	Унікальний ID, маршрут, клієнт, параметри перевезення
Оптимізований маршрут	Список точок маршруту, відстань, передбачений час
Призначений ресурс	Водій і ТЗ відповідно до параметрів замовлення
Візуалізація	Графічне представлення маршруту у вікні користувача
Повідомлення користувачу	Інформація про збереження, помилки, валідацію
Звіт	Структурований текстовий файл (PDF/текст), придатний до друку

Архітектура майбутньої системи повинна бути модульною, що дозволить підтримувати гнучкість і масштабованість реалізації. Передбачено реалізацію таких логічних компонентів: GUI-модуль, модуль логіки обробки замовлень, маршрутизатор, модуль візуалізації (на основі бібліотеки JGraphX або GraphStream), підсистема роботи з базою даних.

З огляду на технічні параметри (див. пункт 1.2, таблиця 1.4), реалізацію системи рекомендовано виконувати з дотриманням шаблону проектування MVC (Model-View-Controller), що дозволить ізольовано обробляти логіку, інтерфейс та дані. Це також забезпечить підтримку тестування окремих компонентів та майбутнє розширення системи – наприклад, шляхом підключення GPS-модуля або інтеграції API для онлайн-моніторингу вантажів.

Узагальнена постановка задачі забезпечує повну специфікацію майбутньої системи, формує вимоги до структури вихідного коду та визначає контрольні точки реалізації, які буде застосовано на наступному етапі – при створенні концептуальної моделі.

1.5 Моделювання предметної області

Моделювання предметної області є ключовим етапом процесу розробки інформаційної системи, який дозволяє формалізувати вимоги до програмного забезпечення, ідентифікувати взаємодії між суб’єктами системи, визначити основні функціональні блоки та сформувати архітектуру майбутнього продукту.

У рамках даного проєкту було застосовано стандартизовану нотацію UML (Unified Modeling Language), яка дозволяє створювати формальні графічні представлення компонентів, об’єктів, процесів та їх зв’язків [16], [22].

Предметна область охоплює діяльність логістичної компанії, пов’язану зі створенням, обробкою та виконанням замовлень на доставку вантажів, керуванням транспортними засобами, маршрутизацією, а також взаємодією між користувачами різного рівня доступу (адміністратор, оператор логістики, клієнт). Визначення ролей та їхніх дій у системі є основою для побудови діаграми прецедентів, яка демонструє зовнішню поведінку системи з точки зору кінцевих користувачів.

На рис. 1.1 представлено діаграму прецедентів інформаційної системи управління логістичною компанією. У ній виділено три основних актори: Адміністратор, Оператор логістики та Клієнт. Кожен із них взаємодіє з певними підсистемами. Адміністратор має доступ до функцій керування користувачами, транспортом та перегляду звітів. Оператор логістики створює та опрацьовує замовлення, призначає транспорт і формує маршрути. Клієнт, у свою чергу, має можливість переглядати історію замовлень та отримувати сповіщення (рис. 1.5).

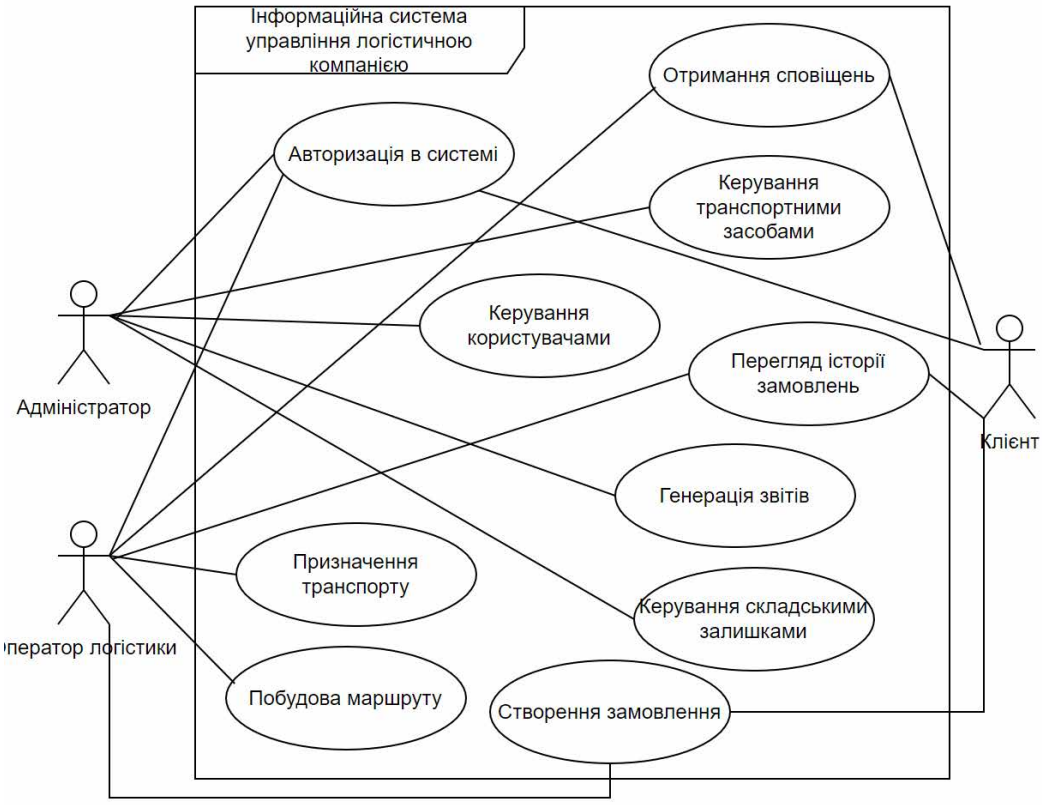

Рис. 1.5 – Діаграма прецедентів інформаційної системи управління логістичною компанією

Для детального опису внутрішньої логіки функціонування інформаційної системи управління логістичною компанією було створено діаграму послідовності (Sequence Diagram), яка ілюструє часову послідовність повідомлень між об’єктами системи під час виконання одного з ключових бізнес-процесів – створення та маршрутизації логістичного замовлення. Цей тип діаграми є ефективним інструментом для відображення динаміки взаємодій у системі та дозволяє формалізувати обмін повідомленнями між компонентами відповідно до принципів об’єктно-орієнтованого проєктування [16], [22].

На рис. 1.6 наведено типовий сценарій взаємодії між суб’єктами системи в рамках процесу створення нового замовлення.

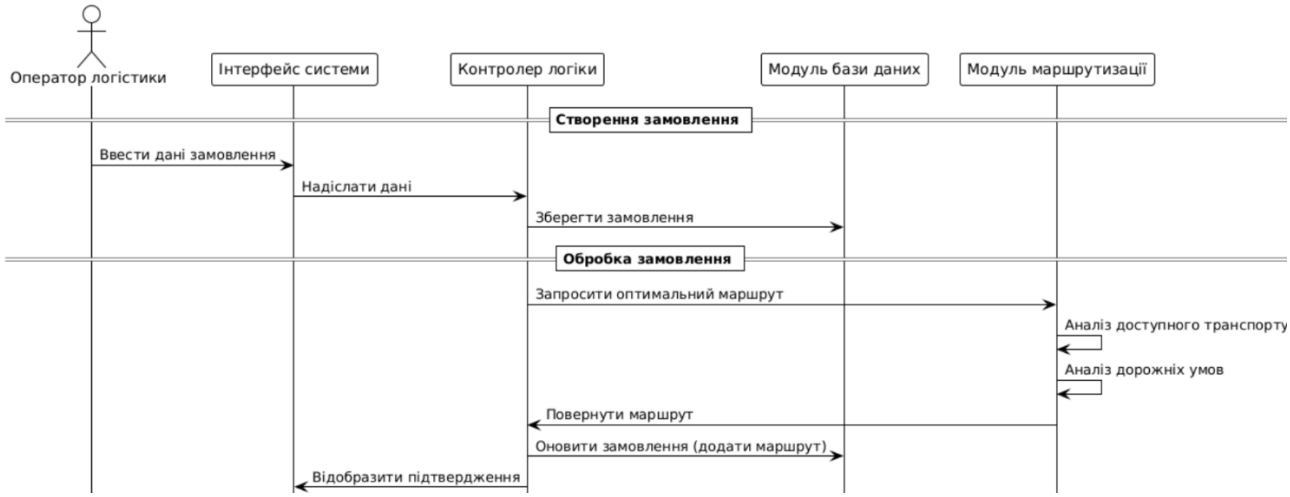

Рис 1.6 - Діаграма послідовності: створення та обробка замовлення

Ініціатором процесу виступає Оператор логістики, який здійснює первинне введення даних через графічний інтерфейс користувача (UI). Інтерфейс передає введену інформацію до контролера логіки (Controller), що виступає посередником між представленням і бізнес-логікою. Контролер перевіряє дані на коректність і ініціює запит на збереження об'єкта замовлення у модулі бази даних (DB), реалізованому за допомогою вбудованої реляційної СУБД.

Після підтвердження успішного збереження системою виконується наступний логічний крок – розрахунок оптимального маршруту доставки. Для цього контролер надсилає запит до окремого компонента – модуля маршрутизації (Routing Module), який виконує аналіз на основі вхідних параметрів (місце відправлення, місце призначення, вага вантажу, тип транспорту, тимчасові обмеження). Всередині цього модуля запускаються два паралельних підпроцеси:

– Аналіз доступного транспорту (відбір вільних ТЗ, що відповідають вимогам замовлення);

– Аналіз дорожніх умов (стан доріг, затори, погодні умови, обмеження руху) [6], [17].

Результати цих підпроцесів консолідуються для побудови оптимального маршруту доставки з урахуванням мінімізації витрат часу та ресурсів. Після формування маршруту модуль маршрутизації повертає згенеровану інформацію

назад до контролера логіки, який здійснює оновлення запису замовлення в БД, додаючи маршрут, призначеного водія та транспортний засіб. Останнім кроком є передача підтвердження про успішне створення та обробку замовлення назад до UI з подальшим відображенням повідомлення для оператора.

Для деталізації алгоритмічної логіки обробки замовлення в системі та ілюстрації умовних переходів між етапами було розроблено діаграму активності (Activity Diagram). На рис. 1.7 представлено логічну схему, що описує дії двох основних учасників процесу – Оператора логістики та Системи. Діаграма побудована із використанням нотації UML із розподілом на свімлейни, що дозволяє відобразити зону відповідальності кожного суб'єкта в рамках спільного процесу [16].

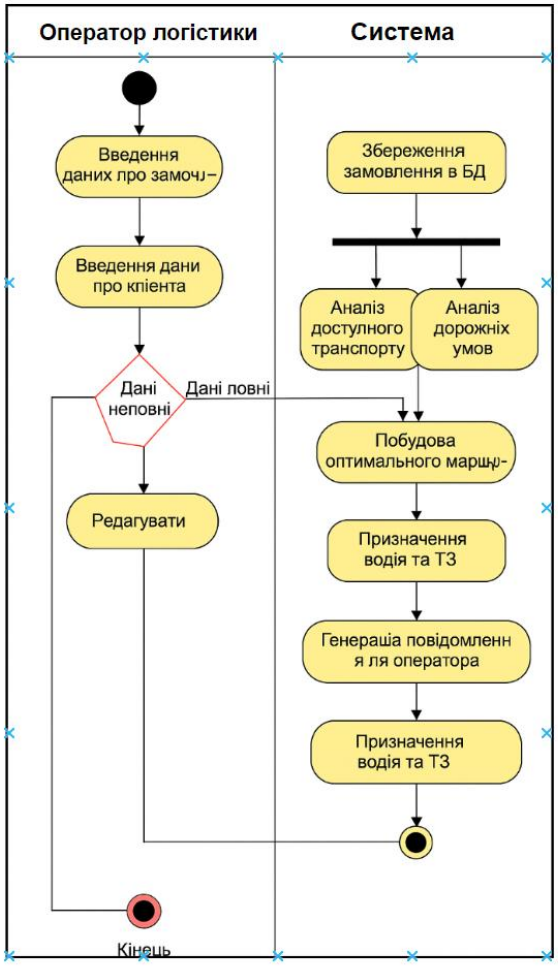

Рис 1.7 – Діаграма активності: процес обробки замовлення логістичної компанії

Процес починається з введення даних про замовлення та даних клієнта, які здійснюються оператором. На цьому етапі реалізовано умовну розвилку, що дозволяє визначити повноту введеної інформації. Якщо дані виявляються неповними або некоректними, система ініціює перехід до редагування форми введення, після чого оператор має можливість доповнити чи виправити інформацію. У разі повноти даних процес переходить до автоматизованої обробки.

Система виконує збереження замовлення в базі даних, після чого активуються два паралельних потоки:

– Аналіз доступного транспорту, який перевіряє поточний парк ТЗ і їхню доступність;

– Аналіз дорожніх умов, що включає API-запити до зовнішніх джерел (наприклад, служб моніторингу трафіку або погодних систем).

Результати з двох гілок обробки інтегруються, і система виконує побудову оптимального маршруту, після чого ініціюється модуль призначення водія та ТЗ. Додатково генерується службове повідомлення для інтерфейсу користувача з підтвердженням виконаних дій. Завершальним етапом є фіксація кінцевого стану – успішне завершення процесу створення замовлення.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Проєктування логічної моделі бази даних є критичним етапом у процесі розробки інформаційної системи управління логістичною компанією, оскільки саме на цьому рівні здійснюється структуризація предметної області у вигляді сутностей, їхніх атрибутів та зв’язків між ними. Логічна модель даних формує концептуальну основу, яка відображає реальні процеси, що відбуваються у логістичній діяльності, включаючи управління транспортом, складським обліком, маршрутами доставки, клієнтськими замовленнями та іншими ключовими аспектами. Вона не лише задає основу для подальшої реалізації фізичної структури бази даних, а й забезпечує уніфікований підхід до обробки інформації, що дозволяє ефективно автоматизувати бізнес-процеси. Також логічна модель сприяє нормалізації даних, усуненню надлишковості, зменшенню ризиків виникнення аномалій при оновленні або вилученні інформації, та забезпечує підтримку цілісності даних у межах всієї системи. Завдяки цьому досягається висока надійність, масштабованість і узгодженість функціонування інформаційної системи компанії.

Для формалізації логічної структури даних використано ER-модель (Entity–Relationship), яка побудована відповідно до класичної нотації Чена з графічним поданням сутностей у вигляді прямокутників, атрибутів – у вигляді списків у межах сутності, а зв’язків – у вигляді ромбів з позначенням кардинальностей. Логічна модель представлена на рис. 2.1.

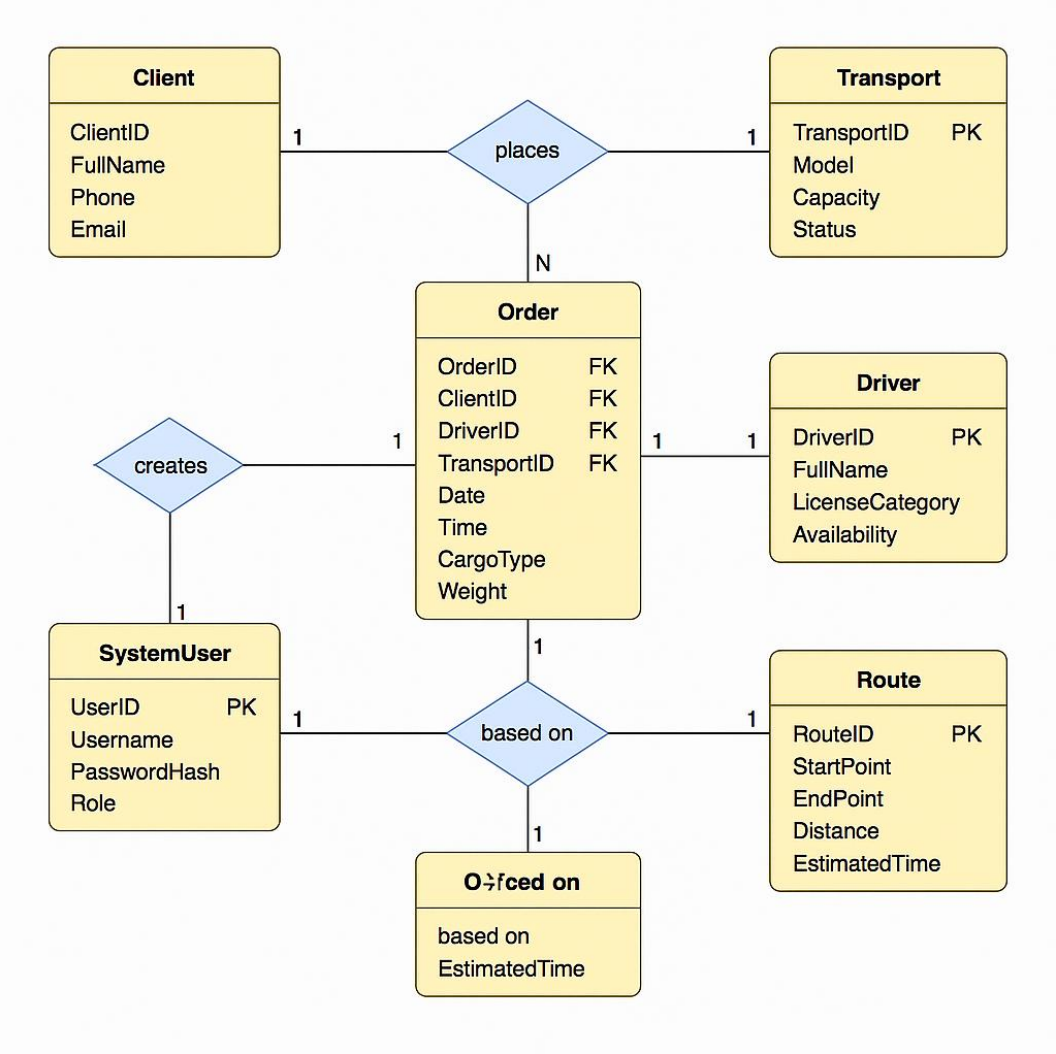

Рис. 2.1 – ER-діаграма логістичної інформаційної системи

У моделі виділено шість ключових сутностей: Client, Order, Transport, Driver, Route та SystemUser, кожна з яких описує окрему логічну складову предметної області.

У структурі логічної моделі центральне місце займає сутність Order, яка акумулює основні бізнес-дані, що відображають процес виконання логістичних операцій. Вона містить унікальний ідентифікатор OrderID, а також атрибути Date, Time, CargoType, Weight та зовнішні ключі ClientID, DriverID, TransportID і RouteID, що забезпечують інтеграцію з іншими сутностями моделі. Усі зв’язки реалізовано з кардинальністю один до одного, що гарантує однозначну відповідність між записами та спрощує логіку обробки запитів у базі даних.

Сутність Client, пов’язана з Order через зв’язок places, забезпечує збереження інформації про замовників логістичних послуг. Вона включає атрибути ClientID, FullName, Phone та Email, і бере участь у зв’язку з кардинальністю 1:N, що означає можливість формування кількох замовлень одним клієнтом.

Функціонально важливою є також сутність SystemUser, яка репрезентує адміністративну частину системи і реалізує авторизацію та рольовий контроль доступу. Ця сутність містить UserID, Username, PasswordHash, Role і формує зв’язок із Order через відношення creates, що відображає створення замовлень адміністратором або оператором.

Сутність Transport зберігає дані про транспортні ресурси, які залучаються до виконання логістичних завдань. Вона включає атрибути TransportID, Model, Capacity, Status і логічно зв’язана з Order, де кожне замовлення асоціюється з одним конкретним транспортним засобом.

Сутність Driver фіксує інформацію про водіїв, які залучаються до перевезень, зокрема DriverID, FullName, LicenseCategory та Availability. Водій, як і транспортний засіб, асоціюється з конкретним замовленням за зв’язком один до одного, що виключає дублювання під час одночасної обробки активних перевезень.

Окрему роль у моделі відіграє сутність Route, яка визначає параметри логістичного маршруту: RouteID, StartPoint, EndPoint, Distance, EstimatedTime. Її зв’язок з Order реалізовано через атрибут EstimatedTime, що використовується як ключове планувальне значення. Такий підхід дозволяє формувати оптимальні шляхи доставки відповідно до заданих обмежень часу та відстані.

На основі цієї моделі даних побудовано таблицю 2.1, у якій представлено узагальнені характеристики сутностей і типи атрибутів, що підлягають реалізації на рівні фізичної бази даних.

Таблиця 2.1 – Основні сутності логічної моделі даних

Сутність	Первинний ключ	Ключові атрибути	Зовнішні ключі
Client	ClientID	FullName, Phone, Email	–
Order	OrderID	Date, Time, CargoType, Weight	ClientID, DriverID, TransportID, RouteID

Продовження таблиці 2.1

Transport	TransportID	Model, Capacity, Status	–
Driver	DriverID	FullName, LicenseCategory, Availability	–
Route	RouteID	StartPoint, EndPoint, Distance, EstimatedTime	–
SystemUser	UserID	Username, PasswordHash, Role	–

Представлена логічна модель задовольняє принципи нормалізації до третьої нормальної форми (3НФ), що виключає надмірність даних і забезпечує цілісність зв’язків між сутностями. Кардинальності між сутностями визначені з урахуванням реальних умов логістичного процесу, що дозволяє уникнути аномалій під час оновлення, вставлення чи видалення даних.

2.2 Діаграма класів і кооперації

Проєктування архітектури програмного забезпечення вимагає чіткого уявлення про структуру класів та їхню взаємодію в рамках ключових бізнес-процесів, що забезпечує логічну та ефективну організацію програмного коду. У даній інформаційній системі для управління логістичними процесами така структура реалізована за допомогою UML-діаграми класів, яка формалізує основні компоненти системи, включаючи сутності, атрибути, методи та зв’язки між об’єктами. Ця діаграма дозволяє зрозуміти, як програмні компоненти системи співвідносяться із реальними бізнес-процесами. Додатково застосовано дві діаграми кооперації, що деталізують механізми взаємодії об’єктів у ключових сценаріях – процесах створення замовлення та формування маршруту доставки. Ці діаграми допомагають проаналізувати логіку обміну повідомленнями між об’єктами, що є критичним для забезпечення узгодженості й надійності програмної реалізації. Таке моделювання сприяє глибшому розумінню

внутрішньої логіки системи, полегшує подальшу розробку, тестування та супровід програмного забезпечення.

Діаграма класів, зображена на рис. 2.2, демонструє статичну структуру основних об’єктів системи та їхніх методів.

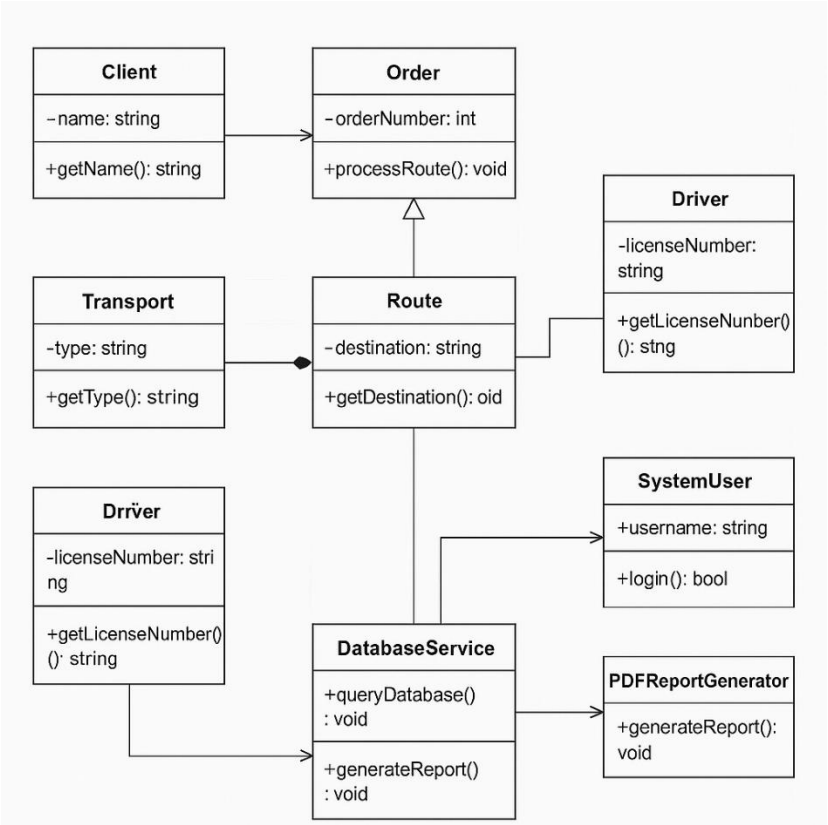

Рис. 2.2 – Діаграма класів проектованої системи

Центральним елементом є клас Order, який інкапсулює основну бізнес-логіку обробки логістичного замовлення і реалізує метод processRoute(). До атрибутів цього класу належить orderNumber:int, що ідентифікує замовлення. Клас пов’язаний асоціативно з класом Client, який зберігає інформацію про замовника (name:string) та надає доступ через метод getName(). Аналогічним чином реалізовано класи Driver і Transport, які зберігають інформацію про водіїв і транспортні засоби. Особливістю є композиційний зв’язок між Route та Transport, який вказує на нерозривну залежність маршруту від типу обраного транспорту. Клас Route зберігає пункт призначення (destination:string) та реалізує метод getDestination(). Додатково в моделі включено службові компоненти DatabaseService та PDFReportGenerator, які відповідають відповідно за

збереження даних і генерацію звітів. Усі методи мають чітко визначену сигнатуру, а поля – відповідний рівень доступу, що забезпечує інкапсуляцію і модульність.

Динамічна взаємодія між класами в межах сценарію створення замовлення відображена на діаграмі кооперації (рис. 2.3).

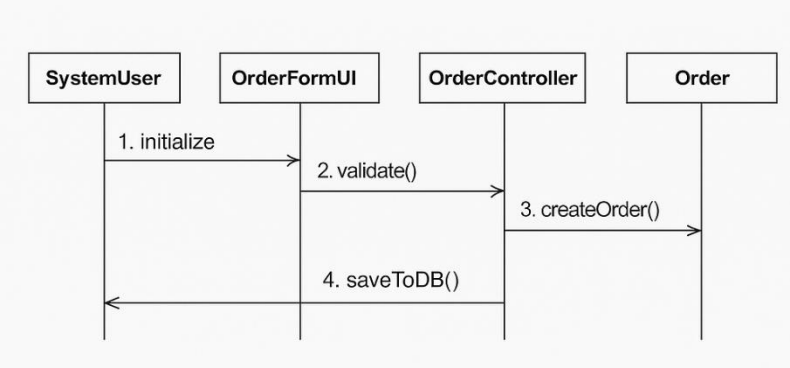

Рис. 2.3 – Діаграма кооперації: сценарій створення логістичного замовлення

Вихідним об’єктом виступає **SystemUser**, який ініціалізує процес введення даних через інтерфейс користувача **OrderFormUI**. Цей компонент відповідає за відображення форми та збирання необхідної інформації для створення замовлення. Після завершення введення користувачем, викликається метод **validate()**, який здійснює перевірку коректності введених параметрів, таких як формат даних, обов’язковість полів та узгодженість значень. У разі успішної валідації, **OrderController** – контролер, що керує логікою обробки даних – формує об’єкт замовлення шляхом виклику методу **createOrder()**, а далі ініціює його збереження до бази даних за допомогою методу **saveToDB()**. Уся ця послідовність дій демонструє реалізацію архітектурного шаблону **MVC (Model-View-Controller)**, який передбачає чітке розмежування відповідальностей між компонентами системи: **View** (інтерфейс користувача), **Controller** (керування логікою взаємодії) та **Model** (структура та обробка даних). Такий підхід не лише забезпечує гнучкість архітектури, а й спрощує тестування, масштабування та підтримку програмного забезпечення, дозволяючи швидко адаптуватися до змін у бізнес-вимогах.

Другий сценарій взаємодії стосується побудови маршруту доставки на основі вхідних параметрів замовлення (рис. 2.4).

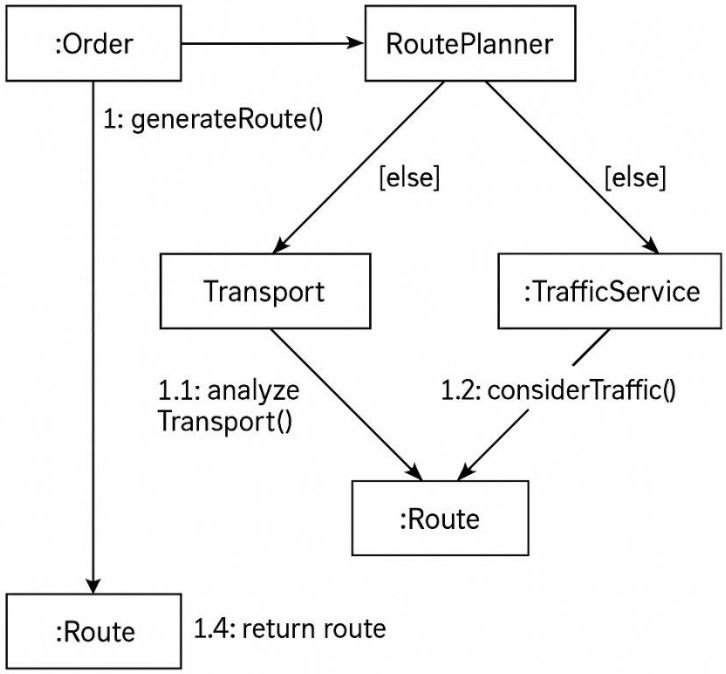

Рис. 2.4 – Діаграма кооперації: сценарій побудови маршруту доставки

Об’єкт Order викликає метод generateRoute() у модулі RoutePlanner. Даний компонент реалізує координацію двох підпроцесів: analyzeTransport() в об’єкті Transport і considerTraffic() в об’єкті TrafficService. Результати обох процедур об’єднуються у вигляді об’єкта Route, який передається назад до замовлення. Така кооперація дозволяє гнучко адаптувати логіку маршрутизації до змінних параметрів середовища: технічних характеристик транспорту, стану дорожньої інфраструктури та обмежень за часом доставки. Обидва підпроцеси позначено умовними переходами [else], що формалізує наявність вибору залежно від ситуації.

Узагальнюючи, діаграма класів забезпечує структурну основу системи, тоді як діаграми кооперації наочно демонструють, як ці компоненти взаємодіють у межах конкретних функціональних процесів. Таке поєднання моделей дозволяє забезпечити логічну цілісність архітектури, формалізувати комунікацію між

об'єктами та створити передумови для масштабованої та підтримуваної реалізації програмного продукту.

2.3 Діаграма пакетів

З метою структурного поділу компонентів програмного забезпечення та формалізації міжмодульних залежностей в інформаційній системі управління логістичною компанією було побудовано діаграму пакетів. Дана діаграма є важливою частиною логічної архітектури, оскільки дозволяє розмежувати функціональні зони відповідальності, забезпечити модульність коду та полегшити супровід і масштабування системи. На рівні UML-нотації діаграма відображає залежності типу <<uses>> між пакетами та розподіл класів відповідно до їхньої логіки.

На рис. 2.5 подано діаграму пакетів, яка включає основні модулі системи: ui, controller, model, service, auth, repository, util.

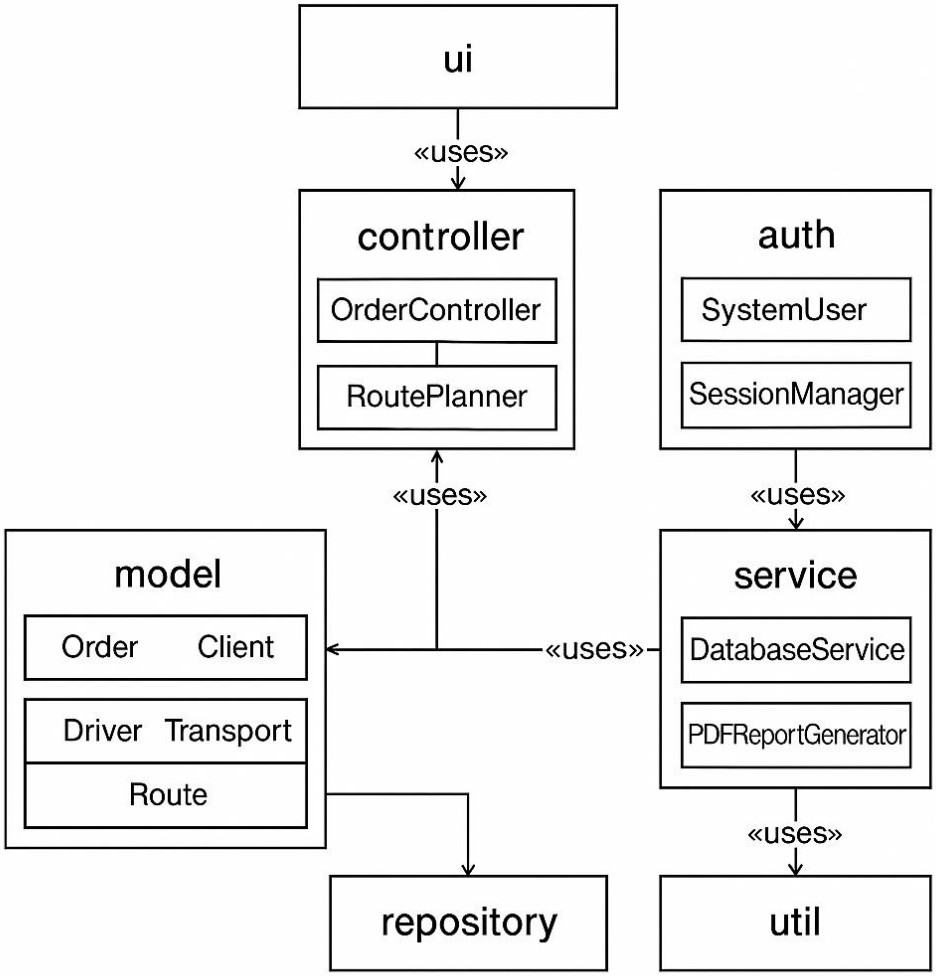

Рис.2.5 – Діаграма пакетів розроблювальної системи

Найвищий рівень ієрархії займає пакет ui, що реалізує графічний інтерфейс користувача та взаємодіє виключно з контролерами, згідно з принципом інверсії залежностей. Пакет controller містить ключові класи бізнес-логіки – OrderController та RoutePlanner, які координують обробку замовлень і формування маршрутів. Цей пакет використовує об’єкти з пакета model, у якому зосереджені основні сутності предметної області: Order, Client, Driver, Transport, Route. Пакет auth забезпечує автентифікацію та управління сесіями за рахунок класів SystemUser і SessionManager, при цьому він також залежить від сервісної логіки. У свою чергу, пакет service реалізує допоміжну функціональність – роботу з базами даних (DatabaseService) та генерацію звітів (PDFReportGenerator). Комунікація з базою даних здійснюється через окремий пакет repository, що забезпечує ізоляцію механізмів доступу до збережених

даних. Пакет util містить допоміжні компоненти, такі як конвертери, валідаційні механізми, шаблони форматування тощо.

Структурну деталізацію компонентів кожного пакета наведено у табл. 2.2, яка узагальнює логіку і відповідальність функціональних груп.

Таблиця 2.2 – Структура та функціональне призначення пакетів

Назва пакета	Вміст (класи)	Призначення
ui	–	Відображення графічного інтерфейсу користувача
controller	OrderController, RoutePlanner	Обробка подій, координація дій користувача
model	Order, Client, Driver, Transport, Route	Модель предметної області, основні логістичні сутності
auth	SystemUser, SessionManager	Реалізація авторизації, перевірка прав доступу
service	DatabaseService, PDFReportGenerator	Допоміжні сервіси для роботи з даними та генерації звітів
repository	–	Доступ до джерел даних, збереження та вибірка інформації з БД
util	–	Утилітарні функції, валідатори, трансформери даних

Діаграма реалізує ізольованість шарів та прямолінійність залежностей, що відповідає архітектурному патерну "чиста архітектура" (Clean Architecture) у спрощеній формі. Це дозволяє зменшити зв’язність між модулями, покращити тестованість компонентів, а також спростити повторне використання та масштабування системи за рахунок чітко визначених меж відповідальності кожного пакета.

2.4 Діаграма компонентів

З метою представлення фізичної структури програмного забезпечення та формалізації зв’язків між основними функціональними модулями системи управління логістичною компанією було побудовано UML-діаграму компонентів. Цей тип діаграми дозволяє візуалізувати логічне розділення

програмної системи на ізольовані частини – компоненти, які реалізують конкретні функціональні можливості, такі як обробка замовлень, управління маршрутами, робота з базами даних або інтерфейсом користувача. Крім того, діаграма визначає інтерфейси взаємодії між компонентами, що дозволяє чітко окреслити зони відповідальності, спростити інтеграцію та покращити обслуговування системи.

Побудована архітектура базується на принципах тришарової моделі, яка включає: презентаційний рівень (відповідає за взаємодію з користувачем, відображення даних та прийом вхідної інформації), рівень прикладної логіки (містить бізнес-правила та координує обробку запитів), рівень доступу до даних і службових сервісів – забезпечує збереження, отримання та обробку даних у базі даних або зовнішніх сервісах).

Такий архітектурний підхід дозволяє досягти ізольованості модулів, низького рівня зв’язності між компонентами та високої масштабованості, що є особливо важливим для складних та динамічних логістичних систем, які потребують постійного розширення функціоналу та адаптації до нових умов. На рис. 2.6 подано діаграму компонентів системи.

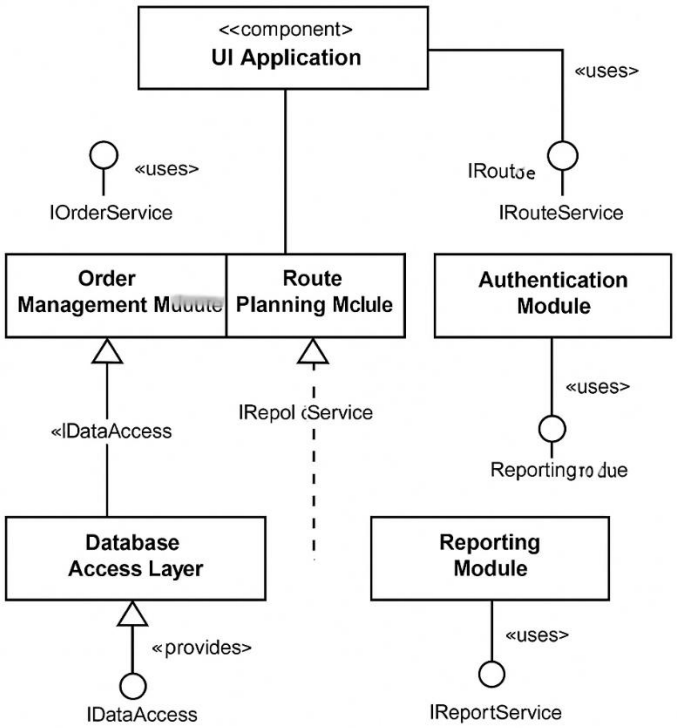

Рис. 2.6 – Діаграма компонентів програмної системи управління логістичною компанією

Презентаційний рівень представлений компонентом UI Application, що реалізує графічний інтерфейс користувача і взаємодіє з основними модулями логіки через стандартизовані інтерфейси IOrderService та IRouteService. Основна бізнес-логіка представлена двома ключовими модулями – Order Management Module і Route Planning Module, які координують створення та обробку логістичних замовлень, а також побудову маршрутів. Обидва компоненти використовують інтерфейс IDataAccess для взаємодії з нижчим рівнем – Database Access Layer, який інкапсулює запити до СУБД і є єдиним постачальником доступу до даних. Усі запити до бази виконуються через чітко визначений контракт IDataAccess, що формально реалізований у нижньому компоненті (<<provides>>).

Компонент Authentication Module забезпечує механізми автентифікації та керування сесіями користувачів. Він реалізує власну логіку авторизації та використовує інтерфейс IReportService, що належить до Reporting Module. Останній модуль відповідає за формування вихідної документації (зокрема звітів у PDF-форматі) та може бути повторно використаний у будь-якому іншому сервісі системи. Структурну деталізацію компонентів, їх призначення та зв’язки подано в табл. 2.3.

Таблиця 2.3. – Опис основних компонентів інформаційної системи

Назва компонента	Інтерфейси	Призначення
UI Application	IOrderService, IRouteService	Взаємодія з користувачем, ініціація процесів логістики
Order Management Module	IDataAccess	Обробка замовлень, створення об’єктів доставки
Route Planning Module	IDataAccess, IReportService	Побудова маршрутів, облік обмежень, інтеграція з модулем звітів
Database Access Layer	IDataAccess	Запити до БД, збереження та вибірка даних
Authentication Module	IReportService	Авторизація, реєстрація сесій, доступ до служб звітів

Reporting Module	IReportService	Формування PDF-звітів, експорт результатів

Зображена структура демонструє високий рівень модульності, що дозволяє проводити ізольоване тестування кожного компонента, спрощує внесення змін у логіку окремих підсистем без впливу на загальну архітектуру, а також створює передумови для масштабування системи шляхом додавання нових сервісів або розширення існуючих інтерфейсів. Таке моделювання сприяє формуванню надійної основи для реалізації принципів SOLID у розробці компонентно-орієнтованого програмного забезпечення.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління базою даних

У процесі розробки інформаційної системи управління логістичною компанією ключову роль відіграє вибір системи управління базою даних (СУБД), яка забезпечує надійне зберігання, доступ, оновлення та фільтрацію логістичних даних. З урахуванням вимог до гнучкості, автономності, швидкості розробки, простоти обслуговування, а також відсутності потреби в масштабованому реляційному середовищі було прийнято рішення використовувати файл-орієнтований підхід до зберігання даних з використанням формату JSON (JavaScript Object Notation) як базової структури даних системи.

Застосування формату JSON замість класичних реляційних СУБД (MySQL, PostgreSQL тощо) обумовлено низкою факторів. По-перше, структура об’єктів предметної області є вкладеною, змінною за складом, що добре відповідає схемі зберігання у вигляді документів. По-друге, система має працювати в умовах локального середовища без встановлення повноцінного серверного ПЗ, що забезпечує швидке розгортання, незалежність і мінімальні системні вимоги. По-третє, вибір JSON дозволяє напряму серіалізувати/десеріалізувати об’єкти у вигляді файлів, що значно спрощує інтеграцію з клієнтською частиною, написаною на JavaScript або Node.js.

Структура бази даних організована у вигляді набору файлів JSON, кожен з яких відповідає окремій сутності предметної області. Наприклад, окремо зберігаються списки користувачів, замовлень, маршрутів, транспортних засобів і водіїв. Кожен запис є незалежним JSON-об'єктом, а файл виконує функцію колекції. Запити на читання, додавання, оновлення або видалення реалізуються засобами мови програмування (наприклад, через Node.js-фреймворк), зчитуючи та змінюючи дані у структурі пам’яті перед збереженням у файл.

У табл. 3.1 наведено перелік логічних сутностей, їхнє призначення та відповідні файли зберігання у форматі JSON.

Таблиця 3.1 – Логічна структура зберігання даних у JSON-файлах

Сутність	Назва файлу	Призначення
Замовлення	orders.json	Зберігання інформації про логістичні замовлення
Клієнти	clients.json	Дані про замовників, контактна інформація
Водії	drivers.json	Перелік водіїв, категорії прав, доступність
Транспорт	transport.json	Моделі, вантажопідйомність, статус транспортних засобів
Маршрути	routes.json	Географічні точки, відстані, час доставки
Користувачі системи	users.json	Авторизація, ролі, дані для входу
Повідомлення	messages.json	Повідомлення, отримані з контактної форми
Звіти	reports.json	Метаінформація про сформовані PDF-звіти (опціонально)

Застосування JSON-файлів забезпечує достатній рівень гнучкості, надає можливість легко редагувати структуру без зміни схеми бази даних, підтримує вкладені об’єкти та масиви, що особливо корисно для маршрутів і замовлень. Крім того, з огляду на те, що система не передбачає великого навантаження та високої транзакційності, відсутність механізмів блокування або індексування не створює критичних обмежень.

3.2 Розробка інформаційної бази

Інформаційна база системи управління логістичною компанією реалізована у вигляді фізичної моделі даних, структурованої за принципом зберігання в окремих JSON-файлах, кожен з яких виконує роль логічно ізольованої колекції. Проте, для підвищення формалізації моделі на етапі проєктування та перевірки цілісності структури була побудована еквівалентна реляційна схема у вигляді SQL-опису. Це дозволило чітко визначити первинні ключі, типи полів, зовнішні зв’язки між сутностями, а також підготувати модель до можливої міграції до повноцінної СУБД у майбутньому.

На рис. 3.1 подано ілюстрацію фізичної моделі даних, яка складається з восьми основних сутностей: користувачі, клієнти, замовлення, водії, транспорт, маршрути, повідомлення та звіти. Кожна з них зберігається у відповідному файлі з розширенням .json. У моделі позначено типи полів (string, number, boolean), первинні ключі (PK), а також логічні зв’язки між сутностями, які відповідають зовнішнім ключам у класичних реляційних БД.

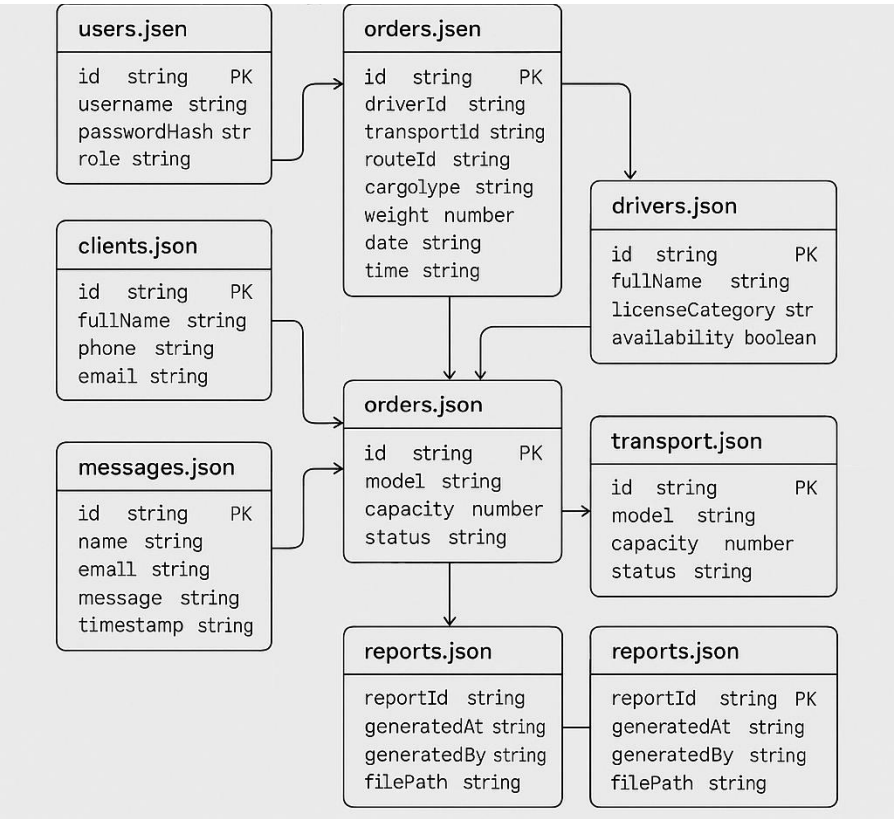

Рис. 3.1 – Фізична модель даних для системи управління логістикою у форматі JSON

Для формального опису структури інформаційної бази нижче наведено фрагменти SQL-коду, які демонструють логічну відповідність між полями JSON-структури та таблицями реляційної моделі. Це дозволяє забезпечити еквівалентність опису, реалізувати валідацію типів та структури при зчитуванні даних у пам’ять застосунку. Лістинг запитів представлений на рис.3.2.

```
CREATE TABLE users (
  iD TEXT PRIY
  username TEXT NOT NULL,
  passwordHash TEXT NOUL,
  role TEXT NOT NULL.
)

CREATE TABLE clients (
  id TEXT PRIY,
  clientId TEXT NOT NULL,
  driverId TEXT,
  transportId TEXT
  routeId TEXT
)

CREATE TABLE orders (
  id TEXT PRIY,
  clientId
  driverId
  transportId TEXT,
  routeId TEXT
  cargpType TEXT,
  weight NUMERIC,
  date TEXT
  time TEXT
  FOREIGN KEY (clients(id),
  FOREIGN KEY references(
  FOREIGN KEY drivers(id),
  FOREIGN KEY transport(id)
  FOREIGN KEY routes(id)
)

CREATE TABLE messages (
  id TEXT PRIY
  name TEXT
  email TEXT
  mestaqp TEXT
)
```

```
CREATE TABLE drivers (
  id TEXT
  model TEXT
  capacity NUMERIC,
  status TEXT
)

CREATE TABLE routes (
  id TEXT
  startPoint TEXT
  endPoint TEXT
  distance NUMERIIC,
  estimatedTime TEXT
)

CREATE TABLE messages (
  id TEXT
  name TEXT
  email TEXT
  message TETXT
  timestamp TEXT
)

CREATE TABLE reports (
  reportId PRIY
  generatedAt TEXT
  generatedBy TEXT
  filePath TEXT
)

CREATE TABLE reports (
  reportId PRIY
  generatedAt TEXT
  generatedBy TEXT
  filePath TEXT
)
```

Рис.3.2 – Лістинг коду sql-запитів для створення таблиць

Кожна таблиця відповідає окремій сутності в структурі предметної області. Наприклад, таблиця orders об'єднує в собі зовнішні ключі до інших таблиць (clientId, driverId, routeId, transportId), що у форматі JSON реалізується через вкладені посилання або логічне зв’язування об’єктів за ідентифікатором. Подібна SQL-репрезентація не використовується безпосередньо під час виконання застосунку, однак є корисним інструментом проєктування, тестування, конвертації або перенесення на інші СУБД (наприклад, PostgreSQL, SQLite).

Розроблена інформаційна база даних поєднує в собі гнучкість та легкість роботи з JSON-структурами у виробничому середовищі з чіткою формалізованою схемою даних, яку можна відобразити у вигляді SQL-моделі для валідації або подальшої міграції. Такий підхід забезпечує як адаптивність системи, так і її відповідність принципам структурного моделювання.

3.3 Архітектура програмного забезпечення

Архітектура інформаційної системи управління логістичною компанією реалізована відповідно до принципів багаторівневої побудови програмного забезпечення, що дозволяє забезпечити модульність, ізоляцію функцій, спрощену підтримку, а також гнучкість масштабування окремих компонентів. Програмна структура організована у вигляді чотиришарової логіки взаємодії: інтерфейсного рівня, рівня бізнес-логіки, сервісного рівня та рівня збереження даних.

На рис. 3.3 представлено архітектурну схему системи.

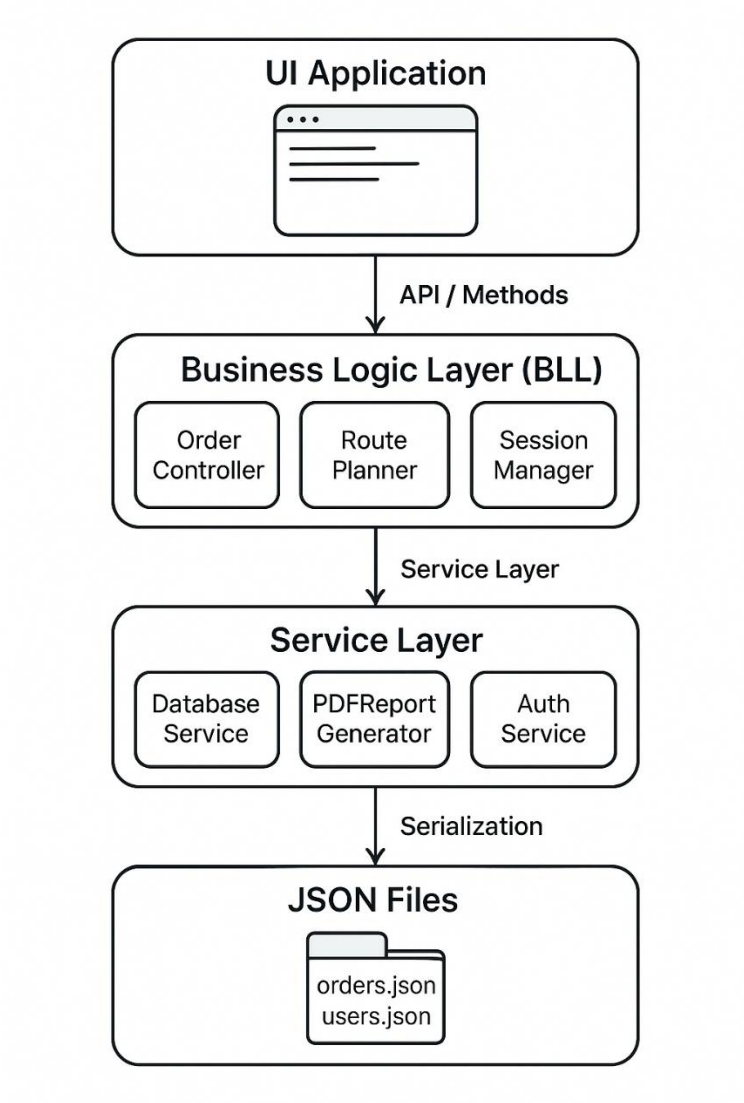

Рис. 3.2 – Архітектура програмної системи управління логістичною компанією

Вона починається з верхнього шару – UI Application, який відповідає за взаємодію з користувачем. Цей компонент реалізує відображення інтерфейсу, обробку подій і виклики функцій за допомогою API або методів. Наступним є Business Logic Layer (BLL), який забезпечує обробку бізнес-процесів. Тут розміщено основні керуючі модулі: Order Controller (керування замовленнями), Route Planner (формування маршрутів), Session Manager (керування сеансами користувачів).

Під рівнем логіки розташовано Service Layer, де зосереджено допоміжні сервіси, необхідні для виконання бізнес-операцій: Database Service (доступ до даних), PDFReport Generator (формування звітності), Auth Service (перевірка автентичності користувача). Взаємодія між шарами реалізується через чітко визначені методи/інтерфейси, що дозволяє забезпечити слабке зв’язування між компонентами. Нарешті, на нижньому рівні архітектури розташовано JSON Files – файлову систему для зберігання структурованих даних у форматі .json, де розміщуються файли orders.json, users.json та інші, відповідно до структури моделі даних.

Для детальнішого розкриття структури взаємодії між компонентами архітектури наведено узагальнену таблицю 3.2 призначення шарів та їхніх основних елементів.

Таблиця 3.2 – Опис функціональних рівнів архітектури

Рівень	Компоненти	Призначення
Презентаційний (UI)	UI Application	Взаємодія з користувачем, відправлення запитів
Логічний (BLL)	Order Controller, Route Planner, Session Manager	Реалізація логіки обробки замовлень, маршрутів і сеансів
Сервісний (Services)	Database Service, PDFReport Generator, Auth Service	Допоміжна логіка для доступу до даних, автентифікації та звітності
Даних (Storage)	orders.json, users.json, інші JSON-файли	Збереження структурованих об'єктів системи у файловій формі

Подана архітектура дозволяє досягти високого рівня ізоляції функцій, спрощує супровід програмного забезпечення, а також забезпечує можливість

швидкої адаптації до змін середовища виконання або розширення функціональності системи без порушення цілісності її логічної структури. Завдяки чітко визначеним межам між шарами та використанню формату JSON як сховища, система зберігає гнучкість і простоту в реалізації без втрати логічної структурованості.

3.4 Вибір інструментарію для створення прикладного програмного забезпечення

У процесі розробки прикладного програмного забезпечення для інформаційної системи управління логістичною компанією критично важливо обґрунтовано обрати інструменти, що забезпечать функціональність, стабільність, зручність інтерфейсу та легкість обслуговування. Основними критеріями відбору виступали: підтримка компонентного підходу, можливість побудови графічного інтерфейсу, сумісність із файловим зберіганням у форматі JSON, доступність бібліотек, кросплатформеність і низькі вимоги до розгортання.

З огляду на вказані вимоги, основним середовищем розробки обрано Java, а для реалізації графічного інтерфейсу – бібліотеку Swing, яка є частиною стандартного пакета Java і дозволяє створювати інтерактивні десктопні застосунки. Swing забезпечує підтримку широкого набору компонентів (таблиць, форм, діалогових вікон, кнопок), можливість обробки подій, гнучке компонування елементів інтерфейсу та підтримку тем оформлення.

Обробка даних реалізується за допомогою стандартних засобів мови Java: зчитування і запис JSON-файлів виконується з використанням бібліотеки org.json або Gson, яка забезпечує двосторонню серіалізацію об'єктів. Файлова система виконує роль сховища – структура даних відповідає логіці сутностей системи (`orders.json`, `users.json`, тощо), що забезпечує простоту пошуку, модифікації та імпорту/експорту.

У табл. 3.3 наведено перелік інструментальних засобів, що використовуються на кожному функціональному рівні програмного забезпечення.

Таблиця 3.3 – Використані інструменти для розробки систем

Компонент системи	Інструментарій	Призначення
Клієнтська частина (UI)	Java Swing	Побудова графічного інтерфейсу, форми, таблиці, елементи керування
Логіка програми (BLL)	Java SE 17, об’єктно-орієнтоване моделювання	Реалізація функціоналу: замовлення, маршрути, автентифікація
Звітність	iText / Apache PDFBox	Генерація звітів у форматі PDF на основі даних із JSON-файлів
Сховище даних	Файлова система, JSON-файли	Структуроване зберігання об'єктів предметної області
Серіалізація	Gson / org.json	Перетворення об'єктів Java у JSON і навпаки
Середовище розробки	IntelliJ IDEA / Eclipse	Створення, налагодження, компіляція та запуск програми
Менеджер залежностей	Maven (опціонально)	Керування бібліотеками, зокрема PDF та JSON

Використання Java у поєднанні з Swing дозволяє створити кросплатформений застосунок із класичною структурою класів, чітким розділенням обов’язків між модулями та можливістю масштабування. Відсутність потреби у зовнішній СУБД знижує технічні вимоги до інфраструктури, а використання JSON-формату забезпечує простоту обміну даними, їх зберігання і перенесення. Таким чином, обраний інструментарій повністю відповідає функціональним і технічним вимогам до розроблюваної системи.

3.5 Алгоритмізація програмних модулів

На етапі алгоритмізації відбувається формалізація логіки функціонування ключових компонентів прикладного програмного забезпечення у вигляді послідовностей дій. Це дозволяє забезпечити зрозумілу й візуалізовану модель, що є основою для наступної програмної реалізації. У межах даного проєкту були

визначені два основні програмні модулі, для яких побудовано блок-схеми алгоритмів: модуль створення замовлення та модуль побудови маршруту. Їхня логіка відображає базову функціональність системи: обробку вхідних даних, валідацію, збереження, виклики підпроцедур та обробку помилок.

Перший алгоритм (рис. 3.3) реалізує логіку створення нового логістичного замовлення. Ініціація відбувається через графічний інтерфейс користувача, який викликає форму заповнення. Користувач вводить дані щодо клієнта, типу вантажу, маршруту, транспорту та водія. Після цього система виконує валідацію введеної інформації. У разі успішного проходження перевірки формується об'єкт Order, який серіалізується в JSON-формат та зберігається у файл orders.json. У випадку помилки користувач отримує відповідне повідомлення з пропозицією повторного введення. Якщо ж операція завершується без помилок, виводиться підтвердження успішного створення замовлення.

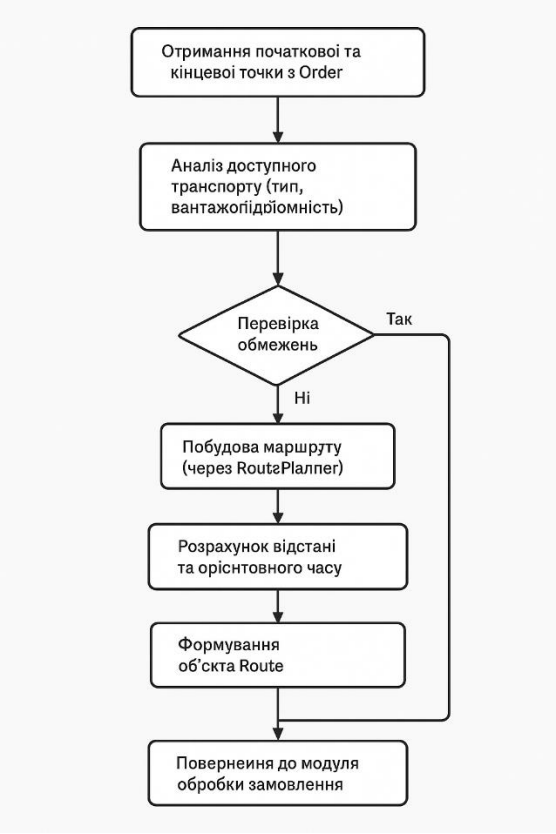

Рис. 3.3 – Блок-схема алгоритму створення нового логістичного замовлення

Другий алгоритм (рис. 3.4) відповідає за побудову маршруту доставки. Алгоритм починається з отримання вихідних координат маршруту (початкова і кінцева точки) з об'єкта Order. Далі виконується аналіз доступного транспорту з урахуванням типу та вантажопідйомності. Після цього проводиться перевірка додаткових обмежень – наприклад, відповідність категорії дороги чи допустима тривалість маршруту. Якщо хоча б одне з обмежень не виконується, система переходить до альтернативного сценарію або інформує користувача. За умови дотримання обмежень ініціюється побудова маршруту через компонент RoutePlanner. Після розрахунку відстані та орієнтовного часу формується об'єкт Route, який повертається до модуля створення замовлення для завершення процесу.

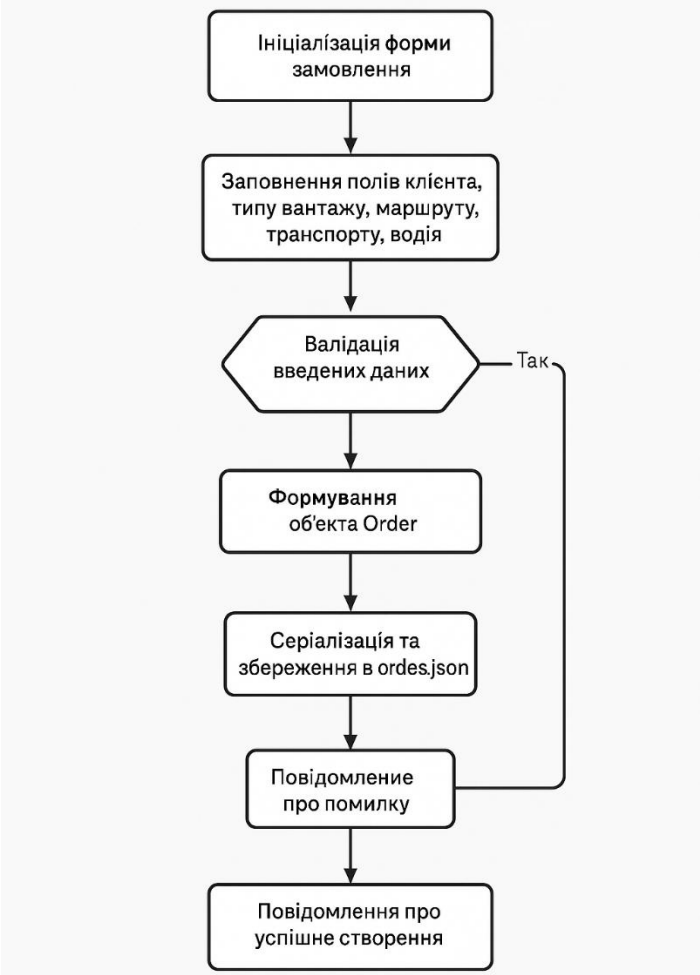

Рис. 3.4 – Блок-схема алгоритму побудови маршруту доставки

Алгоритмізація вищезазначених сценаріїв дозволяє не лише формалізувати поведінку системи, а й слугує еталоном для подальшого

програмного модулювання з використанням Java/Swing. Відповідність між графічною моделлю та програмною реалізацією гарантує передбачувану поведінку компонентів, спрощує налагодження, тестування та впровадження змін у функціонал. Блок-схеми є важливою складовою технічної документації та підтримують якісний перехід від проектного аналізу до безпосереднього програмування.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Тестування системи

Метою тестування є перевірка відповідності розробленої інформаційної системи управління логістичною компанією заявленим функціональним та нефункціональним вимогам. У процесі тестування використовувалися елементи модульного, інтеграційного та системного тестування, що дало змогу виявити помилки на ранніх етапах і оцінити стабільність роботи ключових компонентів системи.

Перший етап тестування передбачав перевірку коректності роботи модуля реєстрації нового користувача. На рисунку 4.1 представлено форму введення імені, електронної пошти, пароля та його підтвердження, а також реалізацію клієнтської валідації введених даних.

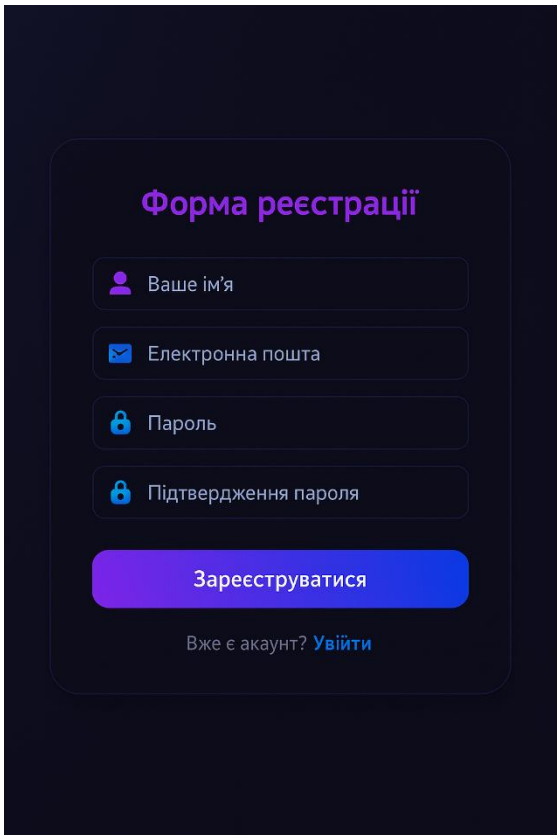

Рис. 4.1 – Форма реєстрації користувача

Наступним етапом перевірки була автентифікація. На рисунку 4.2 показано інтерфейс авторизації користувача, де перевіряється відповідність введеної електронної адреси та пароля даним, збереженим у системі. При введенні некоректних даних система генерує повідомлення про помилку.

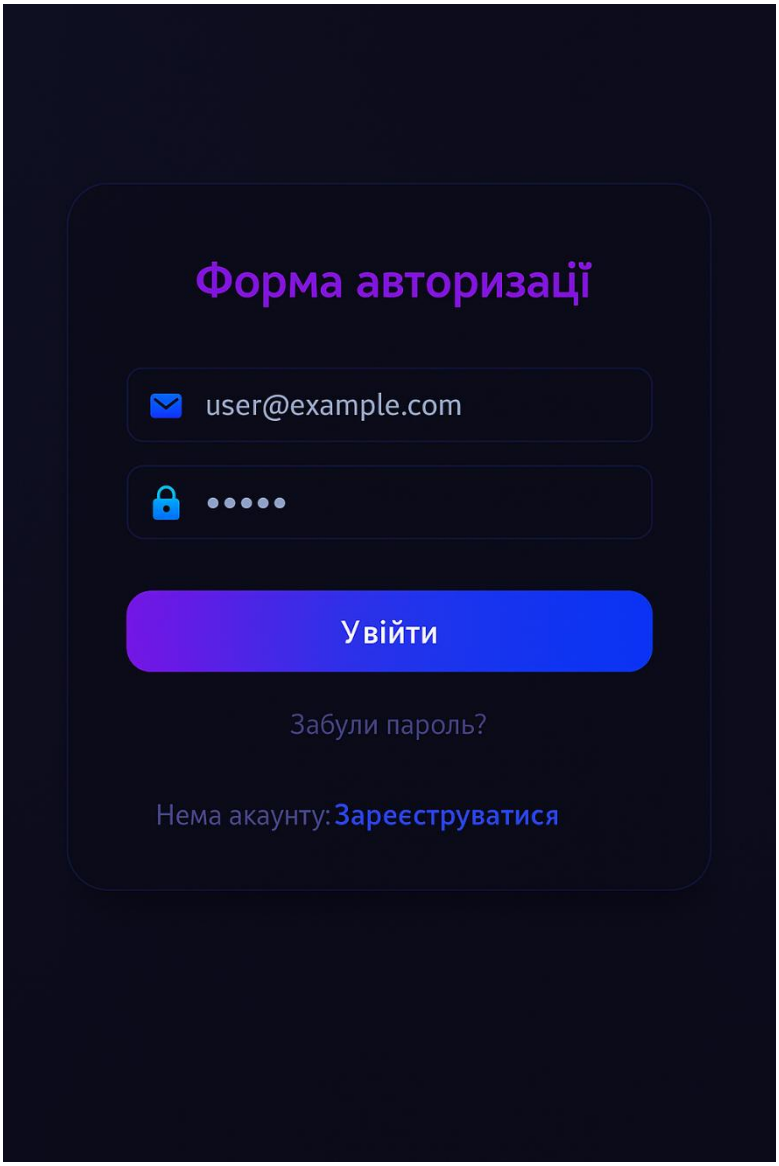

Рис. 4.2 – Форма авторизації користувача

Функціональність обробки замовлень перевірялась шляхом створення логістичного запиту. На рисунку 4.3 зображено форму логістичного замовлення, що включає поля для введення даних про клієнта, маршрут, категорію вантажу, вагу та примітки.

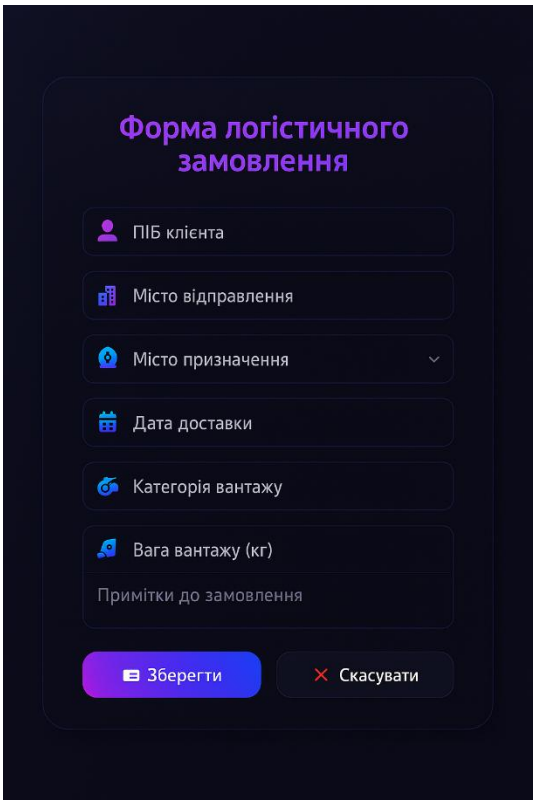

Рис. 4.3 – Форма створення логістичного замовлення

На рисунку 4.4 наведено результати обробки аналітичних даних у вигляді інтерактивної статистики доставок, графіка доходів та загальної інформації про кількість замовлень. Тестування підтвердило коректну візуалізацію даних, що динамічно оновлюються залежно від змін у базі.

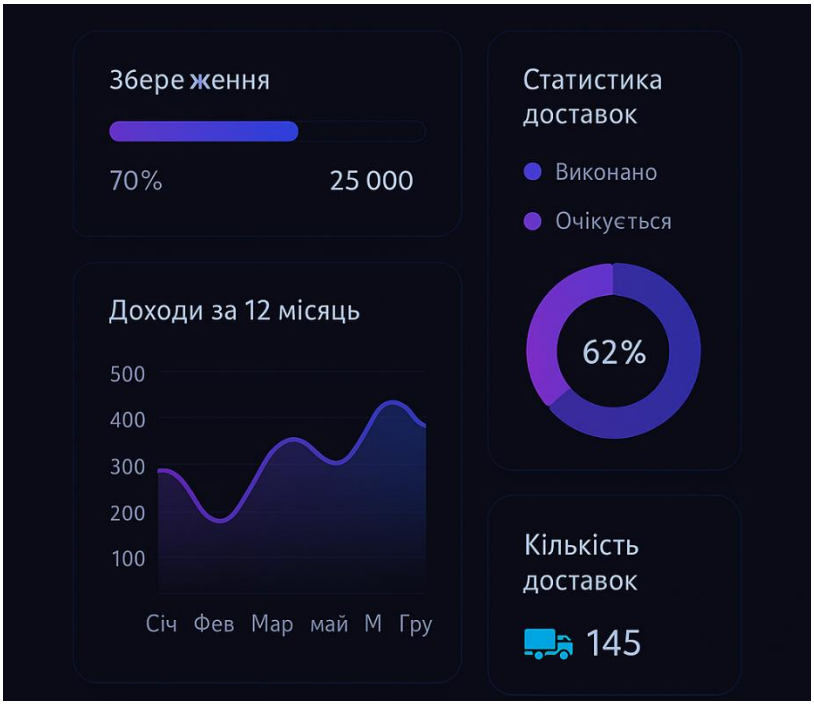

Рис. 4.4 – Аналітична панель логістичних даних

Модуль генерації звітів перевірявся шляхом формування PDF-файлів на основі обраного типу звіту, дати початку та формату. Інтерфейс зображено на рисунку 4.5.

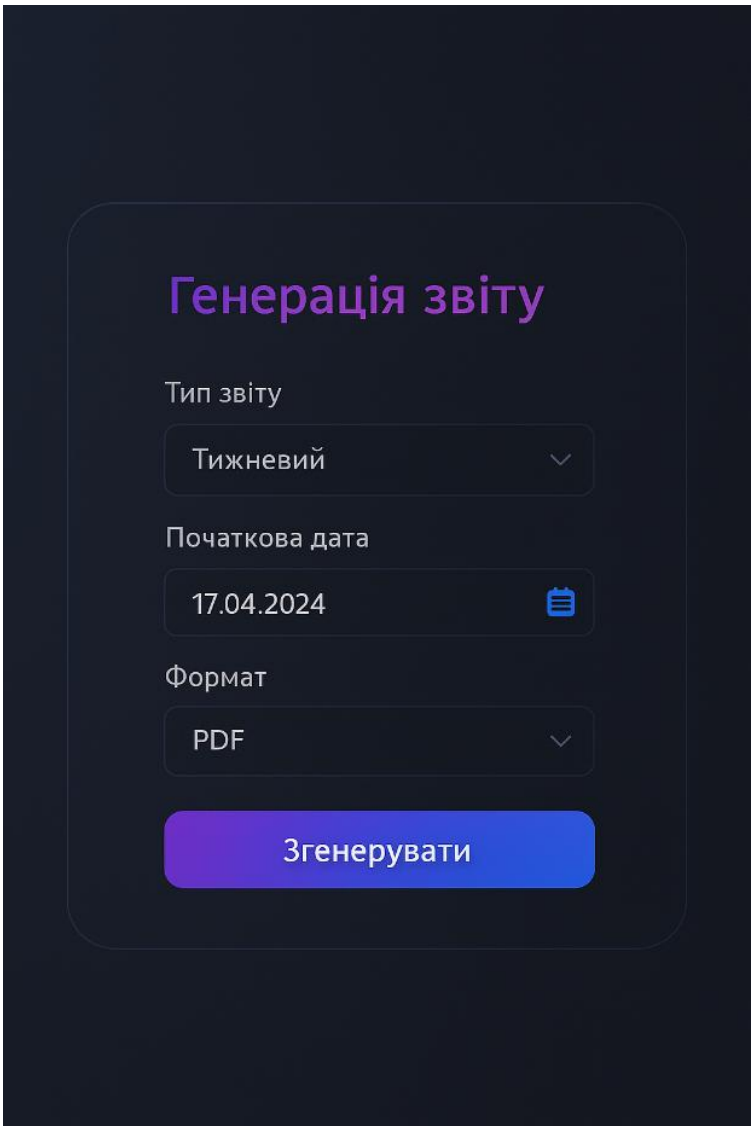

Рис. 4.5 – Форма генерації звіту

Останнім елементом перевірки був модуль історії перевезень. На рисунку 4.6 представлено таблицю, що відображає статуси замовлень у реальному часі, з можливістю додавання нових записів.

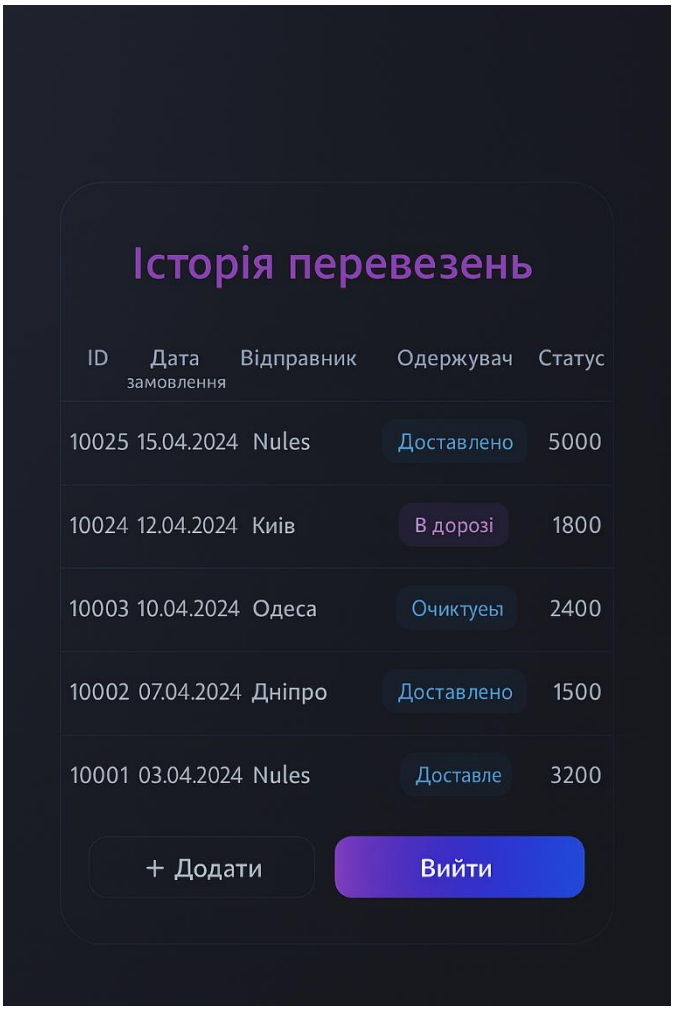

Рис. 4.6 – Інтерфейс історії логістичних перевезень

Усі вищенаведені модулі були протестовані на відповідність функціональним вимогам, що були сформульовані в технічному завданні. Підсумкова таблиця містить результати проведеного тестування (табл. 4.1).

Таблиця 4.1 – Результати тестування функціональних компонентів системи

№	Компонент	Очікувана поведінка	Фактичний результат	Статус
1	Реєстрація користувача	Успішне створення нового облікового запису	Успішно	Пройдено
2	Авторизація	Вхід при коректних даних, повідомлення при помилці	Успішно	Пройдено
3	Створення логістичного замовлення	Збереження замовлення з усіма параметрами	Успішно	Пройдено
4	Візуалізація аналітики	Відображення графіків, статистики	Успішно	Пройдено

Продовження таблиці 4.1

5	Генерація звітів	Створення PDF-звітів відповідно до вибраних параметрів	Успішно	Пройдено
6	Перегляд історії замовлень	Відображення таблиці з фільтрацією та сортуванням	Успішно	Пройдено

Проведене тестування засвідчило функціональну готовність системи до експлуатації. Усі основні сценарії завершилися успішно без фатальних збоїв або критичних помилок. Реалізована структура, побудована на модульному принципі, забезпечила стабільність при навантаженні та спрощення діагностики. Отримані результати свідчать про відповідність системи сформульованим вимогам.

4.2 Вимоги до апаратного та програмного забезпечення

Коректне функціонування програмної системи управління логістичною компанією потребує відповідного технічного середовища, яке забезпечить виконання всіх компонентів із належною продуктивністю, стабільністю та збереженням даних. У цьому підпункті наведено узагальнені вимоги до апаратного забезпечення кінцевих користувачів, а також визначено програмні компоненти, необхідні для повноцінної роботи системи.

На рисунку 4.7 представлено UML-діаграму розгортання, яка демонструє фізичну структуру розміщення компонентів програмної системи у локальному середовищі.

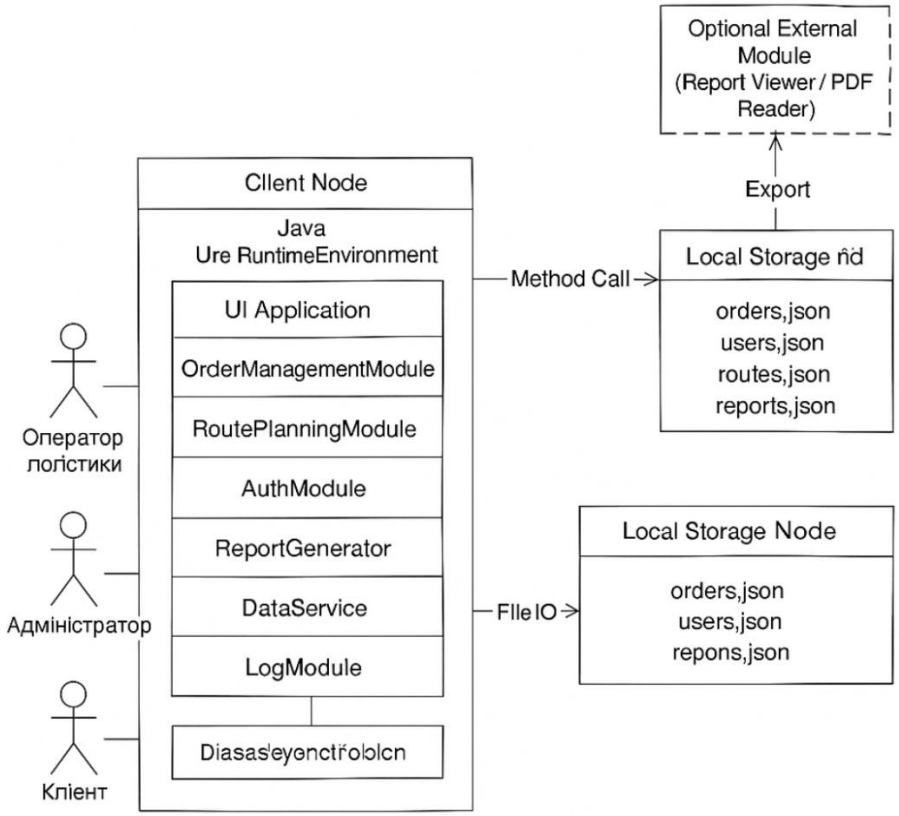

Рис. 4.7 – Діаграма розгортання програмної системи управління логістичною компаніє

Основним вузлом виступає Client Node, який містить інтерпретатор Java та всі програмні компоненти: UI Application, OrderManagementModule, RoutePlanningModule, AuthModule, ReportGenerator, DataService та LogModule. Взаємодія з локальним файловим сховищем (Local Storage Node) здійснюється через канал File IO, який використовується для зчитування та запису даних у форматі JSON (orders.json, users.json, routes.json, reports.json). Крім того, передбачено опціональний зовнішній модуль – PDF Reader, який використовується для перегляду згенерованих звітів. Актори системи – оператор логістики, адміністратор та клієнт – взаємодіють із клієнтською програмою через інтерфейс користувача. Така структура забезпечує автономну роботу системи в умовах локального середовища без залучення серверної інфраструктури.

Апаратне середовище визначено з урахуванням характеру задач, що виконуються системою, а саме: обробка замовлень, збереження даних у JSON-

файлах, генерація PDF-звітів та візуалізація аналітичної інформації. Підсумкові вимоги подано в таблиці 4.2.

Таблиця 4.2 – Мінімальні апаратні характеристики клієнтської машини.

Компонент	Мінімальні характеристики
Центральний процесор	Двоядерний, частота не менше 2.0 ГГц
Оперативна пам’ять	4 ГБ (рекомендовано 8 ГБ для генерації звітів)
Накопичувач	SSD або HDD від 128 ГБ, вільне місце: щонайменше 500 МБ
Дисплей	Роздільна здатність не нижче 1366×768
Інші пристрої	Клавіатура, миша

Такі параметри забезпечують виконання всіх функцій у локальному середовищі без потреби підключення до зовнішніх серверів або хмарних платформ.

Для забезпечення міжплатформенності та автономності розроблене програмне забезпечення реалізовано на Java із застосуванням бібліотеки Swing для побудови графічного інтерфейсу. Програмні компоненти класифіковано за рівнями архітектури системи. Узагальнений перелік необхідного ПЗ подано в таблиці 4.3.

Таблиця 4.3 – Програмне забезпечення для функціонування системи

Програмний компонент	Версія / характеристика	Призначення
Java SE Development Kit (JDK)	Версія 17 або новіша	Запуск програми, середовище виконання
Java Swing	Стандартна бібліотека у складі JDK	Реалізація GUI
JSON бібліотека	org.json / Gson / Jackson	Серіалізація та десеріалізація даних
PDF-бібліотека	iText або аналогічна	Генерація звітів у PDF-форматі
Засіб перегляду PDF	Будь-який (Adobe Reader, Foxit Reader)	Перегляд звітів (опціонально)
Операційна система	Windows 10/11, Linux (Ubuntu 20.04+), macOS 11+	Сумісність із середовищем Java

Важливою особливістю системи є її автономність, що виключає залежність від сторонніх вебсервісів або зовнішніх API. Дані зберігаються у форматі JSON на локальному накопичувачі, доступ до яких здійснюється через відповідний програмний модуль DataService.

Зазначені вимоги забезпечують ефективну експлуатацію системи в умовах середнього навантаження на робочих станціях логістичних операторів, без потреби у виділеному серверному обладнанні чи спеціалізованому адмініструванні

ВИСНОВКИ

У межах виконаної кваліфікаційної роботи було реалізовано повний цикл проєктування, розробки та тестування інформаційної системи управління логістичною компанією, що функціонує у локальному середовищі та відповідає вимогам підприємств малого та середнього бізнесу.

На першому етапі виконано системний аналіз предметної області, у результаті чого ідентифіковано ключові функціональні вимоги до майбутньої системи, зокрема – необхідність автономної обробки замовлень, побудови маршрутів, управління транспортними засобами, генерації звітів і зберігання логістичних даних у форматі, зручному для швидкого доступу й обробки.

У процесі моделювання створено комплекс UML-діаграм, зокрема діаграми прецедентів, класів, послідовностей, компонентів і розгортання, що дозволили формалізувати структуру та динаміку функціонування системи. Ці моделі стали основою для подальшої реалізації архітектури програмного забезпечення.

Проєктування архітектури виконано відповідно до принципів багаторівневої побудови (презентаційний рівень, логіка, сервіси, зберігання), що забезпечило модульність, масштабованість і зручність супроводу. Виділено окремі підсистеми для обробки замовлень, маршрутизації, авторизації користувачів та формування звітів.

Інтерфейс користувача реалізовано на основі бібліотеки Java Swing, що забезпечило платформонезалежність і високу інтерактивність. Було впроваджено адаптивні форми для введення даних, перегляду статистики, генерації звітів та керування історією замовлень.

Інформаційна база побудована у вигляді логічної структури зберігання даних у форматі JSON, що забезпечує простоту обробки, збереження структурованих об’єктів та мінімізацію залежностей від сторонніх СУБД. Крім

того, розроблено еквівалентну SQL-модель, що дозволяє в перспективі здійснити перенесення на реляційну платформу.

У рамках функціонального тестування було перевірено коректність роботи всіх ключових модулів. Тестування засвідчило відповідність реалізованої системи заданим вимогам, її стабільність, цілісність обробки даних та зручність інтерфейсу. Усі сценарії виконувалися без критичних помилок, що підтверджено відповідними результатами.

У підсумку сформульовано рекомендації щодо впровадження розробленої системи в реальні умови підприємства: проведення первинного навчання користувачів, адаптація структури даних під специфіку компанії та можливість масштабування функціоналу, зокрема – інтеграції із системами телематики або GPS-моніторингу.

Усі поставлені в роботі завдання виконано повною мірою, що підтверджує досягнення мети дослідження та створення ефективної, модульної, адаптивної інформаційної системи для логістичного управління.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про транспорт» від 10.11.1994 № 232/94-ВР. – URL: https://zakon.rada.gov.ua/laws/show/232/94-вр

2. Закон України «Про автомобільний транспорт» від 05.04.2001 № 2344-III. – URL: https://zakon.rada.gov.ua/laws/show/2344-14

3. Постанова КМУ № 957 «Про затвердження Порядку ведення обліку логістичних операцій» від 05.11.2012. – URL: https://zakon.rada.gov.ua/laws/show/957-2012-п

4. Єдиний державний реєстр судових рішень. – URL: https://reyestr.court.gov.ua

5. Бондаренко О.І. Інформаційні системи і технології в логістиці. – К.: КНЕУ, 2020. – 312 с.

6. Кузьмін О.Є., Мельник Л.Ю. Логістика: теорія та практика. – Львів: Магнолія 2006, 2019. – 416 с.

7. Балабанова Л.В. Логістика: підручник. – К.: Центр учбової літератури, 2021. – 372 с.

8. Жалдак М.І., Морзе Н.В. Основи інформатики та комп'ютерної техніки. – К.: Освіта, 2021. – 296 с.

9. Летуновський С.О. Об'єктно-орієнтоване програмування на Java. – Харків: Університетська книга, 2022. – 304 с.

10. Чижевський В.П. Розробка ПЗ в середовищі Java Swing. – Чернівці: Рута, 2021. – 228 с.

11. Савінов Ю.І. Java для студентів. – Одеса: НАУ, 2020. – 342 с.

12. Романенко І.В. Моделювання інформаційних систем з використанням UML. – К.: Кондор, 2020. – 240 с.

13. Шапошников І.В. Unified Modeling Language: теорія та практика. – К.: ДІА, 2021. – 214 с.

14. Бровко Ф.Ю. Бази даних: проєктування та реалізація. – Львів: Вид-во ЛНУ, 2020. – 268 с.

15. Кірсанов А.В. SQL у прикладах і задачах. – Х.: ХНУРЕ, 2021. – 192 с.

16. Соммервілл І. Інженерія програмного забезпечення. – К.: Вид. дім «Укрінформнаука», 2018. – 544 с.

17. Кормушин І.В. Технології розробки програмного забезпечення. – К.: НАУ, 2019. – 360 с.

18. Савчук В.М. Архітектура інформаційних систем. – К.: КНУ, 2020. – 380 с.

19. Халін В.В. Проєктування інформаційних систем. – К.: КНУ, 2019. – 352 с.

20. Малахов С.А. Java. Основи програмування. – Х.: Фоліо, 2020. – 280 с.

21. Sommerville I. Software Engineering. – 10th ed. – Pearson, 2016. – 792 p.

22. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. – 2nd ed. – Addison-Wesley, 2005. – 496 p.

23. Horstmann C.S. Core Java Volume I – Fundamentals. – 12th ed. – Pearson, 2022. – 928 p.

24. Eckel B. Thinking in Java. – 4th ed. – Prentice Hall, 2006. – 1150 p.

25. Hohpe G., Woolf B. Enterprise Integration Patterns. – Addison-Wesley, 2004. – 736 p.

26. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Addison-Wesley, 2003. – 208 p.

27. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. – Addison-Wesley, 1994. – 395 p.

28. Kay J. Java Swing. – O'Reilly Media, 2003. – 1250 p.

29. Oracle. Java Swing Tutorial. – URL: https://docs.oracle.com/javase/tutorial/uiswing

30. ISO/IEC 19501:2005. Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2. – ISO, 2005.

ДОДАТОК А

Фрагмент коду інтерфейсу користувача на Java Swing

```
package ui;

import javax.swing.*;
        import java.awt.*;

public class LoginForm extends JFrame {
    private JTextField emailField;
    private JPasswordField passwordField;
    private JButton loginButton;

    public LoginForm() {
        setTitle("Авторизація користувача");
        setSize(400, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new GridBagLayout());

        JLabel emailLabel = new JLabel("Електронна пошта:");
        JLabel passwordLabel = new JLabel("Пароль:");
        emailField = new JTextField(20);
        passwordField = new JPasswordField(20);
        loginButton = new JButton("Увійти");

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);

        gbc.gridx = 0; gbc.gridy = 0;
        add(emailLabel, gbc);

        gbc.gridx = 1; gbc.gridy = 0;
        add(emailField, gbc);

        gbc.gridx = 0; gbc.gridy = 1;
        add(passwordLabel, gbc);

        gbc.gridx = 1; gbc.gridy = 1;
        add(passwordField, gbc);

        gbc.gridx = 1; gbc.gridy = 2;
        add(loginButton, gbc);

        loginButton.addActionListener(e -> {
            String email = emailField.getText();
            String password = new String(passwordField.getPassword());
            // Перевірка логіки авторизації
```

```
            if (email.equals("admin@example.com") &&
password.equals("admin123")) {
                JOptionPane.showMessageDialog(this, "Авторизація успішна.");
            } else {
                JOptionPane.showMessageDialog(this, "Невірні дані.");
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            LoginForm loginForm = new LoginForm();
            loginForm.setVisible(true);
        });
    }
}
```

ДОДАТОК Б

Фрагмент коду обробки замовлень та роботи з JSON-файлом

```
package service;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.*;
        import java.nio.file.Files;
import java.nio.file.Paths;

public class OrderService {

    private static final String FILE_PATH = "orders.json";

    public JSONArray loadOrders() {
        try {
            String content = new
String(Files.readAllBytes(Paths.get(FILE_PATH)));
            return new JSONArray(content);
        } catch (IOException e) {
            System.out.println("Файл не знайдено або порожній. Повертаємо
порожній список.");
            return new JSONArray();
        }
    }

    public void saveOrder(String clientName, String departure, String
destination,
                          String date, String category, double weight) {
        JSONArray orders = loadOrders();

        JSONObject newOrder = new JSONObject();
        newOrder.put("client", clientName);
        newOrder.put("departure", departure);
        newOrder.put("destination", destination);
        newOrder.put("date", date);
        newOrder.put("category", category);
        newOrder.put("weight", weight);
        newOrder.put("status", "Очікується");

        orders.put(newOrder);
```

```
        try (FileWriter file = new FileWriter(FILE_PATH)) {
            file.write(orders.toString(4));
            file.flush();
            System.out.println("Замовлення успішно збережено.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```