КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури»

ВЕРБИЦЬКИЙ ВІТАЛІЙ ОЛЕКСАНДРОВИЧ

(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т. А.

„___” _____ 2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури»

Виконав: студент 4-го курсу, групи КНс-21

Спеціальності: 122 «Комп`ютерні науки»

Спеціалізація: «Інформаційні управляючі системи і технології»

(шифр і назва напряму підготовки, спеціальності)

Вербицкьий В. О.

(прізвище та ініціали)

Керівник к.т.н., доц. Горда О. В.

(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є. Є.

(прізвище та ініціали)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Спеціалізація: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н. Гончаренко Т.А

„ 12 ” лютого 2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Вербицький Віталій Олександрович
Тема роботи: Розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури
затверджена наказом ректора КНУБА № 2650/2 від « 12 » лютого 2024 р.
2. Керівник роботи: Горда Олена Володимирівна, к.т.н, доцент кафедри інформаційних технологій проектування та прикладної математики
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз сучасного стану логістичних інформаційних систем
Р.2. Вимоги для логістичних інформаційних систем
Р.3. Проектування інформаційної системи
Р.4. Реалізація та тестування інформаційної системи
5. Інформаційні слайди:
С.1. Вступ
С.2. Теоретичні основи інформаційних систем в логістиці
С.3. Аналіз вимог до інформаційної системи для логістичних перевезень
С.4. Проектування інформаційної системи

С.5. Реалізація інформаційної системи
С.6. Тестування та впровадження інформаційної системи
С.7. Підсумки роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2024 р.
Р. 2. Алгоритмічне та математичне забезпечення	Лютий 2024 р.
Р. 3. Розробка програмного забезпечення	Березень 2024 р.
Тестовий приклад програми	Квітень 2024 р.
Р. 4. Бізнес план	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	к.т.н., доц. Рябчун Ю.В.		
Прийом програмного продукту	к.т.н., доц. Шабала Є.Є.		

8. Дата видачі завдання: 12 лютого 2024 р.

Керівник _____ Горда О.В.
(підпис) (прізвище та ініціали)

Бакалавр _____ Вербицький В.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Вербицький В.О. Розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури.

Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп'ютерні науки», спеціалізація: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2024.

Бакалаврська робота присвячена дослідженню основних тенденцій, принципів та реалізації інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури. Робота включає аналіз сучасного стану логістичних інформаційних систем, визначення вимог до системи, розробку архітектури та реалізацію прототипу системи. Результати дослідження показали ефективність запропонованих рішень для підвищення продуктивності та оптимізації логістичних процесів.

Ключові слова: клієнт-серверна архітектура, мікросервісна архітектура, логістичні інформаційні системи, оптимізація логістичних процесів, інтеграція технологій, масштабованість.

SUMMARY

Verbitskyi V.O. Development of an information system for organizing logistics transportation using client-server architecture.

Bachelor's thesis for a bachelor's degree in speciality: 122 'Computer Science', specialisation: 'Information management systems and technologies.' - Kyiv National University of Construction and Architecture - Kyiv, 2024.

The bachelor's thesis is devoted to the study of the main trends, principles, and implementation of an information system for organizing logistics transportation using client-server architecture. The work includes an analysis of the current state of logistics information systems, defining system requirements, designing the architecture, and implementing a prototype system. The research results demonstrated the effectiveness of the proposed solutions for increasing productivity and optimizing logistics processes.

Keywords: client-server architecture, microservices architecture, logistics information systems, logistics process optimization, technology integration, scalability.

ЗМІСТ

Зміст 6
Вступ 8
1. АНАЛІЗ СУЧАСНОГО СТАНУ ЛОГІСТИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ 10
1.1. Історичний розвиток логістичних інформаційних систем 10
1.2. Ключові тенденції та виклики у сучасній логістиці 11
1.3. Роль інформаційних технологій у розвитку логістики 13
1.4. Огляд сучасних логістичних інформаційних систем 14
1.5. Використання штучного інтелекту та машинного навчання у логістиці 17
1.6. Вплив Інтернету речей (IoT) на логістичні процеси 18
1.7. Роль великих даних та аналітики у логістиці 20
1.8. Огляд успішних впроваджень логістичних інформаційних систем у різних галузях 22
1.9. Перспективи розвитку логістичних інформаційних систем у майбутньому 24
2. ВИМОГИ ЛОГІСТИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ 27
2.1. Аналіз потреб користувачів та стейкхолдерів 27
2.2. Функціональні вимоги до системи 29
2.3. Нефункціональні вимоги: безпека, продуктивність, масштабованість 32
2.4. Регуляторні вимоги та стандарти 34
2.5. Основні архітектурні шаблони інформаційних систем 36
2.6. Клієнт-серверна архітектура: переваги та недоліки 40
2.7. Розподілені системи: мікросервіси та їх застосування у логістиці 42

2.8. Приклади реалізації логістичних інформаційних систем на основі клієнт-серверної архітектури 45
3. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 49
3.1. Визначення вимог до системи 49
3.2. Аналіз функціональних та нефункціональних вимог 51
3.3. Архітектура системи 53
3.4. Проектування бази даних 53
3.5. Проектування прикладного програмного інтерфейсу (API) 56
3.6. Проектування інтерфейсу користувача 58
4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 61
4.1. Вибір технологій та інструментів для реалізації 61
4.2. Реалізація серверної частини системи 62
4.3. Реалізація клієнтської частини системи 63
4.4. Тестування системи 64
4.5. Аналіз результатів тестування та оптимізація системи 67
Висновки 68
Додаток А: Лістинги коду для серверної частини 70
А.2. Реалізація маршрутів для клієнтів 70
Додаток В: Лістинги коду для клієнтської частини 73

ВСТУП

Сучасний розвиток інформаційних технологій значно впливає на всі сфери діяльності, включаючи логістику. Логістичні інформаційні системи (ЛІС) стають невід'ємною частиною успішного функціонування логістичних процесів, забезпечуючи ефективне управління запасами, планування маршрутів, обробку замовлень та відстеження вантажів. У зв'язку з цим виникає проблема забезпечення максимальної ефективності та надійності ЛІС, яка потребує вирішення.

Основною проблемою, що розглядається у даній роботі, є розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури. Існуючі логістичні системи часто не відповідають сучасним вимогам, мають обмежені можливості для масштабування та інтеграції з новітніми технологіями, такими як штучний інтелект, великі дані та Інтернет речей (IoT). Це знижує їх ефективність та адаптивність до змін у ринкових умовах.

Проблема розробки ефективних ЛІС активно досліджується. Значна кількість наукових робіт присвячена питанням автоматизації логістичних процесів, впровадженню штучного інтелекту та машинного навчання у логістиці, використанню IoT для моніторингу та відстеження вантажів. Проте, існує необхідність у більш детальних дослідженнях, спрямованих на інтеграцію новітніх технологій у ЛІС та розробку систем з високою продуктивністю та безпекою.

Актуальність теми дослідження зумовлена зростаючими вимогами до логістичних інформаційних систем в умовах глобалізації ринків, збільшення обсягів перевезень та необхідності оперативного реагування на зміни попиту. Використання клієнт-серверної архітектури дозволяє створити масштабовану та гнучку систему, яка може легко інтегруватися з новітніми технологіями, забезпечуючи високу продуктивність та безпеку.

Метою даної роботи є розробка інформаційної системи для організації логістичних перевезень з використанням клієнт-серверної архітектури.

Об'єктом дослідження є логістичні інформаційні системи. Предметом дослідження є методи та технології розробки клієнт-серверних логістичних

інформаційних систем, їх інтеграція з новітніми технологіями та оцінка їх ефективності.

Практичне значення даного дослідження полягає у розробці інформаційної системи, яка може бути використана логістичними компаніями для підвищення ефективності управління логістичними процесами. Результати дослідження можуть бути впроваджені у реальних логістичних системах, забезпечуючи оптимізацію маршрутів, зниження витрат та підвищення продуктивності. Апробація результатів дослідження здійснювалась шляхом тестування прототипу системи на базі реальних даних логістичної компанії, що дозволило оцінити її ефективність та відповідність встановленим вимогам.

1. АНАЛІЗ СУЧАСНОГО СТАНУ ЛОГІСТИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1. Історичний розвиток логістичних інформаційних систем

Розвиток логістичних інформаційних систем (ЛІС) почався в середині XX століття з появою перших комерційних комп'ютерів. Перші ЛІС мали на меті автоматизацію окремих процесів, таких як управління запасами та планування виробництва. У 1960-х роках були розроблені системи матеріального планування (MRP), які дозволили підприємствам більш ефективно керувати своїми ресурсами. Ці системи забезпечували точний розрахунок потреб у матеріалах і допомагали зменшити запаси та скоротити час виробництва.

У 1970-х роках розвиток обчислювальної техніки зробив можливим створення більш потужних та надійних ЛІС. З'явилися системи управління складом (WMS) та системи управління транспортом (TMS), які дозволили автоматизувати та оптимізувати логістичні процеси на складах та в транспортних мережах. Ці системи значно підвищили ефективність логістичних операцій, забезпечивши точне управління запасами та оптимізацію маршрутів доставки.

У 1980-х роках інтегровані системи планування ресурсів підприємства (ERP) об'єднали всі ключові бізнес-процеси, включаючи логістику, виробництво, фінанси та управління персоналом. ERP-системи забезпечили комплексний підхід до управління підприємством, що дозволило підвищити ефективність та зменшити витрати. Вони інтегрували дані з різних відділів і дозволили отримувати повну картину діяльності підприємства в режимі реального часу.

У 1990-х роках інтернет і мобільні технології сприяли подальшому розвитку ЛІС. З'явилися системи управління ланцюгами постачання (SCM), які дозволили підприємствам інтегрувати всі етапи ланцюга постачання від постачальника до кінцевого споживача. SCM-системи забезпечили прозорість і контроль на всіх етапах ланцюга постачання, що дозволило значно підвищити ефективність логістичних процесів.

З початку XXI століття ЛІС продовжують еволюціонувати під впливом новітніх технологій, таких як штучний інтелект, машинне навчання, Інтернет речей (IoT) та великі дані. Ці технології дозволяють автоматизувати та оптимізувати логістичні процеси на новому рівні, забезпечуючи більш високу точність, швидкість та ефективність. Сучасні ЛІС стають все більш інтелектуальними та адаптивними, що дозволяє підприємствам швидко реагувати на зміни в ринкових умовах та потребах клієнтів.

Таким чином, історичний розвиток ЛІС демонструє, як технологічні інновації змінюють підходи до управління логістичними процесами, роблячи їх більш ефективними, прозорими та адаптивними до сучасних викликів.

1.2. Ключові тенденції та виклики у сучасній логістиці

Сучасна логістика зазнає значних змін, що впливають на її розвиток і створюють нові виклики для управління. Однією з ключових тенденцій є цифровізація логістичних процесів. Завдяки впровадженню цифрових технологій підприємства можуть автоматизувати та оптимізувати свої операції, що дозволяє знизити витрати та підвищити ефективність. Сучасні інформаційні системи, такі як ERP (Enterprise Resource Planning), WMS (Warehouse Management Systems), TMS (Transportation Management Systems) та SCM (Supply Chain Management), допомагають інтегрувати всі етапи логістичного ланцюга, забезпечуючи прозорість і контроль на кожному рівні. Цифровізація охоплює як великі підприємства, так і малий та середній бізнес, підвищуючи їхню конкурентоспроможність на глобальному ринку.

Ще однією важливою тенденцією є автоматизація та роботизація логістики. Використання роботів на складах, автоматизованих систем пакування та сортування, а також дронів для доставки дозволяє значно підвищити продуктивність і точність виконання операцій. Ці технології не тільки знижують витрати на робочу силу, але й мінімізують помилки, які можуть виникати при ручній обробці вантажів. Завдяки цьому, логістичні компанії можуть забезпечувати вищу якість обслуговування клієнтів і швидше реагувати на їхні потреби.

Використання штучного інтелекту (ШІ) та машинного навчання (МН) також відіграє важливу роль у сучасній логістиці. Ці технології відкривають нові можливості для прогнозування попиту, оптимізації маршрутів, управління запасами та покращення обслуговування клієнтів. Завдяки ШІ та МН компанії можуть аналізувати великі обсяги даних з різних джерел і приймати більш точні та швидкі рішення. Наприклад, прогнозування попиту на основі аналізу історичних даних та поточних тенденцій дозволяє ефективніше планувати запаси і уникати дефіцитів або надлишків товарів.

Інтернет речей (IoT) забезпечує можливість підключення різних пристроїв та сенсорів до мережі, що дозволяє моніторити та керувати логістичними процесами в режимі реального часу. Завдяки IoT компанії можуть відстежувати стан вантажів, умов зберігання, місцезнаходження транспортних засобів і ефективність роботи обладнання. Це значно підвищує прозорість логістичних операцій і дозволяє швидко реагувати на будь-які непередбачені ситуації.

Екологічна складова також стає все більш важливим аспектом логістичних стратегій. Зростаюча увага до екологічних питань змушує компанії шукати шляхи зниження викидів вуглецю, оптимізації маршрутів і впровадження енергоефективних рішень. Використання екологічно чистих транспортних засобів, таких як електромобілі та гібридні вантажівки, а також оптимізація логістичних шляхів дозволяють зменшити негативний вплив на довкілля. Компанії також впроваджують стратегії вторинного використання матеріалів та зменшення відходів, що сприяє створенню більш «чистих» логістичних ланцюгів.

Однак разом із ключовими тенденціями логістика стикається з численними викликами. Глобалізація ланцюгів постачання створює складні та розгалужені мережі, що вимагають високого рівня координації та управління. Забезпечення прозорості та контролю на всіх етапах ланцюга постачання стає дедалі складнішим завданням, особливо у випадках, коли ланцюги постачання охоплюють кілька континентів і різні регуляторні середовища.

Зростаючі очікування споживачів щодо швидкості та якості доставки також створюють значні виклики для логістичних компаній. Сучасні споживачі очікують

не лише швидкої доставки, але й можливості відстежувати посилки в режимі реального часу. Це вимагає від логістичних компаній високого рівня сервісу та швидкої реакції на запити клієнтів. Крім того, компанії повинні забезпечувати гнучкість у виборі методів доставки та можливість здійснення повернень, що додає складності до управління логістичними операціями.

Кібербезпека стає критично важливою у зв'язку з ростом кількості цифрових рішень і підключених пристроїв. Забезпечення захисту даних та систем від кібератак стає першочерговим завданням для логістичних компаній. Витоки даних або атаки на інформаційні системи можуть призвести до значних фінансових втрат і підриву репутації компаній.

Регуляторні вимоги та стандарти, що постійно змінюються, створюють додаткові складнощі для глобальних логістичних операцій. Компанії повинні дотримуватися різних вимог до безпеки, екологічних стандартів і митних процедур у різних країнах. Це вимагає постійного моніторингу регуляторного середовища та адаптації процесів відповідно до нових вимог.

Нарешті, відсутність кваліфікованих кадрів у сфері логістики та інформаційних технологій стає ще одним з викликів для компаній. Швидкий розвиток технологій вимагає нових знань та навичок від працівників, але на ринку праці часто бракує фахівців, що мають релевантний досвід роботи з новітніми системами і технологіями.

Таким чином, сучасна логістика знаходиться на перетині значних технологічних зрушень та викликів, що вимагають нових підходів до управління та розвитку логістичних процесів. Використання передових інформаційних систем та технологій дозволяє компаніям ефективно реагувати на ці виклики та забезпечувати конкурентні переваги на ринку.

1.3. Роль інформаційних технологій у розвитку логістики

Інформаційні технології відіграють ключову роль у розвитку сучасної логістики, забезпечуючи інтеграцію, автоматизацію та оптимізацію логістичних процесів. Завдяки інформаційним технологіям, логістичні компанії можуть ефективніше управляти своїми операціями, знижувати витрати, підвищувати

точність і швидкість доставки, а також забезпечувати високий рівень обслуговування клієнтів.

Однією з основних функцій інформаційних технологій у логістиці є інтеграція всіх ланок логістичного ланцюга. Сучасні інформаційні системи, такі як ERP (Enterprise Resource Planning), WMS (Warehouse Management Systems), TMS (Transportation Management Systems) і SCM (Supply Chain Management), дозволяють об'єднати всі етапи логістичного процесу в єдину інформаційну систему. Це забезпечує прозорість і контроль на всіх рівнях, дозволяючи компаніям отримувати повну картину своїх операцій в режимі реального часу.

Крім інтеграції, інформаційні технології забезпечують автоматизацію логістичних процесів. Автоматизація дозволяє знижувати витрати на робочу силу, мінімізувати помилки, підвищувати продуктивність та забезпечувати високу точність виконання операцій. Наприклад, автоматизовані складські системи можуть виконувати операції з прийому, зберігання та відвантаження товарів з мінімальною участю людини, що дозволяє знижувати витрати та підвищувати продуктивність.

Інформаційні технології також забезпечують оптимізацію логістичних процесів. Оптимізація дозволяє знижувати витрати, скорочувати час виконання операцій, підвищувати точність та забезпечувати високу ефективність. Наприклад, оптимізація маршрутів доставки дозволяє знижувати витрати на транспортування та скорочувати час доставки.

Важливим аспектом є використання аналітичних інструментів та великих даних для прийняття обґрунтованих рішень. Аналітичні системи дозволяють аналізувати великі обсяги даних з різних джерел, виявляти тенденції, прогнозувати попит та оптимізувати логістичні операції. Завдяки цьому, компанії можуть приймати більш обґрунтовані та ефективні рішення, що дозволяє підвищувати продуктивність та конкурентоспроможність.

1.4. Огляд сучасних логістичних інформаційних систем

Сучасні логістичні інформаційні системи значно покращують управління ланцюгами постачання, знижуючи витрати та підвищуючи ефективність операцій.

Вони охоплюють різні аспекти логістики, включаючи управління складом, транспортом та ланцюгами постачання. Розглянемо основні типи таких систем.

1.4.1. Автоматизовані системи управління складом (WMS)

Автоматизовані системи управління складом (Warehouse Management Systems, WMS) є ключовим інструментом для ефективного управління складськими операціями. WMS дозволяють автоматизувати та оптимізувати процеси прийому, зберігання, переміщення та відвантаження товарів. Вони забезпечують точний облік запасів, підвищують продуктивність праці та знижують витрати на складські операції.

Основними функціями WMS є:

Прийом товарів: Системи автоматизують процеси прийому товарів на склад, включаючи перевірку документів, розміщення товарів у відповідні місця зберігання та реєстрацію в системі.

Зберігання: WMS оптимізують розміщення товарів на складі, використовуючи методи управління простором, що дозволяє максимально ефективно використовувати складські площі.

Переміщення товарів: Системи автоматизують процеси переміщення товарів між різними зонами складу, що дозволяє знизити витрати на переміщення та підвищити точність виконання операцій.

Відвантаження: WMS автоматизують процеси підготовки та відвантаження товарів, включаючи комплектацію замовлень, пакування та маркування.

Завдяки WMS компанії можуть значно підвищити точність обліку запасів, знизити витрати на складські операції та підвищити продуктивність праці.

1.4.2. Транспортні інформаційні системи (TMS)

Транспортні інформаційні системи (Transportation Management Systems, TMS) забезпечують автоматизацію та оптимізацію транспортних операцій. Вони допомагають ефективно управляти плануванням маршрутів, управлінням транспортними засобами та водіями, а також відстеженням вантажів у режимі реального часу. TMS дозволяють знизити витрати на транспортування, скоротити час доставки та підвищити точність виконання замовлень.

Основними функціями TMS є:

Планування маршрутів: TMS оптимізують маршрути доставки, враховуючи різні фактори, такі як відстань, вартість транспортування, час доставки та умови дорожнього руху.

Управління транспортними засобами: Системи допомагають ефективно управляти парком транспортних засобів, включаючи їх технічне обслуговування, планування завантаження та відстеження в режимі реального часу.

Відстеження вантажів: TMS забезпечують можливість відстеження вантажів на кожному етапі їх транспортування, що дозволяє компаніям контролювати процес доставки та швидко реагувати на будь-які відхилення від плану.

Завдяки TMS компанії можуть значно знизити витрати на транспортування та скоротити час доставки.

1.4.3. Системи управління ланцюгами поставок (SCM)

Системи управління ланцюгами поставок (Supply Chain Management Systems, SCM) забезпечують інтеграцію та координацію всіх етапів ланцюга постачання від постачальника до кінцевого споживача. Вони дозволяють ефективно управляти попитом, плануванням поставок, управлінням запасами та логістичними операціями. SCM-системи забезпечують високу прозорість та контроль на всіх етапах ланцюга постачання, що дозволяє знижувати витрати, підвищувати точність та швидкість виконання замовлень, а також забезпечувати високий рівень обслуговування клієнтів.

Основними функціями SCM є:

Управління попитом: SCM дозволяють прогнозувати попит на продукцію, враховуючи історичні дані, поточні тенденції та сезонні коливання.

Планування поставок: Системи забезпечують ефективне планування поставок, враховуючи попит на продукцію, наявність запасів та виробничі потужності.

Управління запасами: SCM дозволяють ефективно управляти запасами на всіх етапах ланцюга постачання, що дозволяє уникати дефіцитів або надлишків продукції.

Координація логістичних операцій: Системи забезпечують координацію всіх логістичних операцій, включаючи транспортування, зберігання та обробку вантажів.

Таким чином, сучасні логістичні інформаційні системи, такі як WMS, TMS та SCM, значно покращують управління логістичними процесами. Вони дозволяють компаніям ефективно управляти своїми операціями, знижувати витрати та підвищувати конкурентоспроможність на ринку.

1.5. Використання штучного інтелекту та машинного навчання у логістиці

У сучасній логістиці все більше застосовується штучний інтелект (ШІ) та машинне навчання (МН), які значно покращують ефективність і точність логістичних процесів. Ці технології дозволяють автоматизувати багато рутинних операцій, покращувати прогнозування попиту, оптимізувати маршрути транспортування та забезпечувати більш високу якість обслуговування клієнтів.

Однією з основних переваг використання ШІ та МН у логістиці є можливість покращення прогнозування попиту. Традиційні методи прогнозування часто базуються на історичних даних та простих моделях, які можуть не враховувати поточні тенденції та сезонні коливання. Натомість, ШІ та МН дозволяють аналізувати великі обсяги даних з різних джерел, включаючи ринкові тренди, поведінку споживачів, погодні умови та інші фактори. Це дозволяє створювати більш точні прогнози попиту, що, в свою чергу, допомагає уникнути дефіциту або надлишків товарів.

ШІ та МН також використовуються для оптимізації маршрутів транспортування. Традиційні методи планування маршрутів можуть бути неефективними, особливо в умовах змінних дорожніх умов та високої інтенсивності руху. Використовуючи ШІ, логістичні компанії можуть аналізувати в реальному часі інформацію про дорожню обстановку, враховувати час доставки, витрати на паливо та інші фактори. Це дозволяє створювати оптимальні маршрути, що знижує витрати на транспортування та скорочує час доставки.

Іншою важливою областю застосування ШІ та МН у логістиці є управління запасами. Традиційні методи управління запасами можуть бути неефективними, оскільки вони часто базуються на припущеннях і не враховують реальні потреби ринку. Використовуючи ШІ та МН, компанії можуть аналізувати дані про продажі, попит та інші фактори, щоб автоматично регулювати рівні запасів. Це допомагає уникнути дефіциту або надлишків товарів, знижуючи витрати на зберігання та підвищуючи рівень обслуговування клієнтів.

ШІ та МН також дозволяють покращувати якість обслуговування клієнтів. Наприклад, чат-боти, що використовують ШІ, можуть автоматично відповідати на запити клієнтів, надаючи їм актуальну інформацію про статус їхніх замовлень, умови доставки та інші деталі. Це дозволяє знижувати навантаження на службу підтримки клієнтів та підвищувати рівень обслуговування.

Важливо також зазначити, що ШІ та МН можуть використовуватися для виявлення та запобігання шахрайству в логістиці. Аналізуючи великі обсяги даних, ці технології можуть виявляти підозрілі патерни поведінки та автоматично попереджати про можливі шахрайські дії. Це допомагає забезпечити безпеку логістичних операцій та знизити ризики фінансових втрат.

Таким чином, використання штучного інтелекту та машинного навчання у логістиці відкриває нові можливості для підвищення ефективності, точності та якості логістичних процесів. Ці технології дозволяють автоматизувати рутинні операції, покращувати прогнозування попиту, оптимізувати маршрути транспортування, ефективніше управляти запасами та забезпечувати високий рівень обслуговування клієнтів. Вони також сприяють підвищенню безпеки логістичних операцій та зниженню ризиків шахрайства, що робить логістичні системи більш надійними та стійкими до викликів сучасного ринку.

1.6. Вплив Інтернету речей (IoT) на логістичні процеси

Інтернет речей (IoT) відкриває нові горизонти для логістики, надаючи можливість підключати до мережі різноманітні пристрої та сенсори, що дозволяє значно підвищити ефективність та точність логістичних операцій. IoT впливає на

логістичні процеси на всіх етапах ланцюга постачання, від складування до доставки кінцевому споживачу.

Однією з ключових переваг IoT є можливість моніторингу та відстеження вантажів у режимі реального часу. Використовуючи підключені пристрої та сенсори, компанії можуть отримувати точну інформацію про місцезнаходження товарів, їхній стан та умови транспортування. Наприклад, датчики температури, вологості та удару можуть відстежувати умови зберігання та транспортування чутливих товарів, таких як медикаменти або харчові продукти. Це дозволяє швидко реагувати на будь-які відхилення від норми та вживати необхідних заходів для забезпечення якості продукції.

IoT також значно покращує управління складом. Підключені пристрої можуть автоматично відстежувати наявність товарів на складі, їх переміщення та розміщення. Це дозволяє забезпечити точний облік запасів, знизити ризик помилок та підвищити продуктивність праці. Крім того, автоматизовані системи на основі IoT можуть оптимізувати процеси прийому, зберігання та відвантаження товарів, забезпечуючи більш ефективне використання складських площ та ресурсів.

Ще однією важливою областю застосування IoT є оптимізація транспортних процесів. Підключені транспортні засоби можуть відстежувати маршрути, витрати палива, технічний стан та інші параметри в режимі реального часу. Це дозволяє логістичним компаніям оптимізувати маршрути доставки, знижувати витрати на паливо та обслуговування транспортних засобів, а також підвищувати безпеку перевезень. Наприклад, система може автоматично пропонувати оптимальні маршрути, враховуючи дорожні умови, трафік та погодні умови, що дозволяє зменшити час доставки.

IoT також сприяє покращенню обслуговування клієнтів. Підключені пристрої дозволяють клієнтам відстежувати свої замовлення в режимі реального часу, отримувати повідомлення про статус доставки та очікуваний час прибуття. Це підвищує прозорість логістичних операцій та довіру клієнтів до логістичних компаній. Крім того, можливість отримувати точну інформацію про

місцезнаходження та стан товарів дозволяє компаніям швидко реагувати на запити клієнтів та надавати більш точну і своєчасну інформацію.

Використання IoT також сприяє підвищенню безпеки логістичних операцій. Підключені пристрої можуть автоматично виявляти та попереджати про потенційні загрози, такі як несанкціонований доступ до вантажу, порушення умов транспортування або аварійні ситуації. Це дозволяє знизити ризики втрат і пошкоджень вантажів, а також забезпечити безпеку працівників і транспортних засобів.

Узагальнюючи, вплив Інтернету речей на логістичні процеси є значним і багатогранним. IoT дозволяє забезпечити моніторинг та відстеження вантажів у режимі реального часу, покращує управління складом і транспортними процесами, підвищує прозорість і якість обслуговування клієнтів, а також забезпечує вищий рівень безпеки логістичних операцій. Це робить логістичні системи більш ефективними, точними та стійкими до викликів сучасного ринку.

1.7. Роль великих даних та аналітики у логістиці

У сучасній логістиці великі дані та аналітика відіграють ключову роль у підвищенні ефективності та гнучкості логістичних процесів. Використання великих обсягів даних, які генеруються на різних етапах логістичного ланцюга, дозволяє отримувати глибокі інсайти, приймати обґрунтовані рішення та оптимізувати операції.

Однією з основних переваг використання великих даних у логістиці є можливість покращення прогнозування попиту. Традиційні методи прогнозування часто базуються на обмеженій кількості історичних даних, що може призводити до неточних прогнозів. Застосування великих даних дозволяє враховувати широкий спектр факторів, включаючи сезонні коливання, ринкові тренди, поведінку споживачів та інші змінні. Це дозволяє створювати більш точні прогнози попиту, що допомагає уникати дефіциту або надлишків товарів та забезпечувати ефективне управління запасами.

Аналітика великих даних також значно покращує управління запасами. Аналізуючи дані про продажі, попит, сезонні коливання та інші фактори, логістичні

компанії можуть оптимізувати рівні запасів, знижуючи витрати на зберігання та підвищуючи обслуговування клієнтів. Наприклад, аналітичні системи можуть автоматично виявляти патерни поведінки споживачів та прогнозувати попит на продукцію в різні періоди часу. Це дозволяє компаніям своєчасно поповнювати запаси та уникати дефіцитів або надлишків товарів.

Ще однією важливою сферою застосування великих даних є оптимізація транспортних процесів. Аналітика дозволяє логістичним компаніям аналізувати маршрути, витрати на паливо, час доставки та інші параметри, щоб оптимізувати транспортні операції. Наприклад, аналіз даних про дорожні умови, трафік та погодні умови дозволяє створювати оптимальні маршрути доставки, що знижує витрати на транспортування та скорочує час доставки. Крім того, великі дані дозволяють виявляти неефективності в транспортних процесах та вживати заходів для їх усунення.

Аналітика великих даних також допомагає покращувати обслуговування клієнтів. Аналізуючи дані про поведінку клієнтів, відгуки та запити, логістичні компанії можуть краще розуміти потреби своїх клієнтів та адаптувати свої послуги відповідно до їхніх очікувань. Це дозволяє підвищувати рівень задоволеності клієнтів та забезпечувати більш персоналізоване обслуговування. Наприклад, аналіз відгуків клієнтів може допомогти виявити проблемні моменти в логістичних операціях та вживати заходів для їх виправлення.

Великі дані також відіграють важливу роль у забезпеченні безпеки логістичних операцій. Аналіз даних з різних джерел, включаючи сенсори, відеоспостереження та інші пристрої, дозволяє виявляти підозрілі активності та попереджати про можливі загрози. Це допомагає запобігати крадіжкам, втратам та пошкодженням вантажів, а також забезпечувати безпеку працівників та транспортних засобів.

Таким чином, великі дані та аналітика є невід'ємною частиною сучасної логістики, забезпечуючи глибокі інсайти, точне прогнозування, оптимізацію операцій та покращення обслуговування клієнтів. Використання цих технологій дозволяє логістичним компаніям ефективно управляти своїми процесами,

знижувати витрати, підвищувати продуктивність та забезпечувати високу якість послуг.

1.8. Огляд успішних впроваджень логістичних інформаційних систем у різних галузях

Успішне впровадження логістичних інформаційних систем (ЛІС) у різних галузях свідчить про їхню важливу роль у підвищенні ефективності та конкурентоспроможності підприємств. Нижче розглянемо кілька прикладів успішного застосування ЛІС у різних галузях.

1.8.1. Роздрібна торгівля

У роздрібній торгівлі впровадження логістичних інформаційних систем дозволило значно підвищити ефективність управління запасами та логістичними операціями. Наприклад, компанія Walmart, один із найбільших роздрібних продавців у світі, успішно впровадила систему управління ланцюгами поставок (SCM). Використання цієї системи дозволило Walmart оптимізувати процеси закупівель, знизити витрати на зберігання та транспортування, а також покращити обслуговування клієнтів. Система SCM забезпечує прозорість на всіх етапах ланцюга постачання, що дозволяє швидко реагувати на зміни попиту та уникати дефіциту товарів.

1.8.2. Виробництво

У виробничій галузі ЛІС допомагають оптимізувати процеси планування виробництва, управління запасами та постачання. Компанія Toyota, відомий виробник автомобілів, впровадила систему управління складом (WMS) і систему планування ресурсів підприємства (ERP). Це дозволило Toyota покращити управління запасами комплектуючих, знизити витрати на зберігання та підвищити продуктивність виробничих процесів. Системи WMS та ERP забезпечують точний облік запасів, автоматизують процеси закупівель та виробництва, що дозволяє Toyota забезпечувати високу якість продукції та швидко реагувати на змінні потреби ринку.

1.8.3. Охорона здоров'я

Впровадження ЛІС у галузі охорони здоров'я сприяє покращенню управління постачанням медикаментів та медичного обладнання. Наприклад, компанія McKesson, одна з найбільших фармацевтичних дистриб'юторів у США, впровадила комплексну систему управління ланцюгами поставок. Ця система дозволила McKesson оптимізувати процеси закупівель та постачання, забезпечити точний облік медикаментів та підвищити ефективність логістичних операцій. Використання ЛІС забезпечує прозорість на всіх етапах ланцюга постачання, що дозволяє McKesson швидко реагувати на потреби медичних установ та забезпечувати своєчасне постачання якісних медикаментів.

1.8.4. Електронна комерція

У сфері електронної комерції ЛІС є критично важливими для забезпечення швидкої та точної доставки товарів. Компанія Amazon, один із найбільших онлайн-ритейлерів у світі, успішно впровадила систему управління складом (WMS), транспортну інформаційну систему (TMS) та систему управління ланцюгами поставок (SCM). Це дозволило Amazon автоматизувати процеси прийому, зберігання та відвантаження товарів, оптимізувати маршрути доставки та забезпечити високу прозорість на всіх етапах ланцюга постачання. Використання ЛІС дозволяє Amazon швидко обробляти великі обсяги замовлень, забезпечувати своєчасну доставку та високий рівень обслуговування клієнтів.

1.8.5. Харчова промисловість

У харчовій промисловості ЛІС допомагають забезпечити ефективне управління постачанням та зберіганням продуктів харчування. Компанія Nestlé, один із найбільших виробників харчових продуктів у світі, впровадила систему управління ланцюгами поставок (SCM) та систему управління складом (WMS). Це дозволило Nestlé оптимізувати процеси закупівель, виробництва та постачання, забезпечити точний облік запасів та підвищити ефективність логістичних операцій. Використання ЛІС дозволяє Nestlé забезпечувати високу якість продукції, своєчасну доставку та задовольняти потреби клієнтів у різних регіонах світу.

Узагальнюючи, успішне впровадження логістичних інформаційних систем у різних галузях свідчить про їхню важливу роль у підвищенні ефективності, зниженні витрат та покращенні обслуговування клієнтів. Використання ЛІС дозволяє компаніям автоматизувати та оптимізувати логістичні процеси, забезпечити прозорість на всіх етапах ланцюга постачання та швидко реагувати на змінні потреби ринку. Це робить логістичні системи більш ефективними, гнучкими та стійкими до викликів сучасного ринку.

1.9. Перспективи розвитку логістичних інформаційних систем у майбутньому

Майбутнє логістичних інформаційних систем (ЛІС) обіцяє значні інновації, що сприятимуть підвищенню ефективності та адаптивності логістичних процесів. Технологічний прогрес і зростаючі потреби ринку стимулюють компанії до впровадження передових рішень.

ЛІС дедалі частіше інтегруються з новітніми технологіями, такими як штучний інтелект (ШІ), Інтернет речей (IoT), блокчейн та великі дані. ШІ відкриває нові горизонти для прогнозування попиту, оптимізації маршрутів та управління запасами. Наприклад, аналізуючи великі обсяги даних у реальному часі, системи ШІ можуть швидко адаптуватися до змін у ринкових умовах, забезпечуючи більш точне планування.

IoT значно підвищує рівень моніторингу логістичних процесів. Сенсори, вбудовані в транспортні засоби та складське обладнання, дозволяють відстежувати стан вантажів, умови транспортування та місцезнаходження в режимі реального часу. Це забезпечує прозорість на всіх етапах ланцюга постачання, зменшуючи ризики пошкодження або втрати товарів.

Роботизація логістичних процесів продовжує набирати обертів. Автоматизовані склади, де роботи виконують завдання з прийому, зберігання та відвантаження товарів, вже не є фантастикою. Вони забезпечують високу продуктивність, знижують витрати на робочу силу та мінімізують помилки. Впровадження дронів для доставки товарів також стає все більш популярним. Ці пристрої можуть значно скоротити час доставки, особливо в умовах міського

трафіку або важкодоступних регіонах. Очікується, що в майбутньому дрони стануть важливим елементом логістичних ланцюгів.

Конкуренція на ринку змушує логістичні компанії шукати шляхи для надання більш персоналізованих послуг. Використання великих даних дозволяє глибше розуміти потреби клієнтів і пропонувати рішення, адаптовані до їхніх специфічних вимог. Це може включати налаштування логістичних процесів під індивідуальні потреби клієнтів, що підвищує їхню задоволеність та лояльність.

Усвідомлення екологічних проблем стимулює логістичні компанії впроваджувати стійкі практики. ЛІС відіграють ключову роль у зниженні викидів вуглецю через оптимізацію маршрутів, використання екологічно чистих транспортних засобів та впровадження енергоефективних рішень. Це не тільки допомагає зберегти довкілля, але й покращує репутацію компаній.

Зростання цифровізації збільшує ризик кібератак. ЛІС мають бути оснащені сучасними засобами кібербезпеки, такими як шифрування даних та багатоетапна аутентифікація, щоб захистити інформацію та операції від потенційних загроз. Це є критично важливим для підтримання безпеки та надійності логістичних ланцюгів.

Хмарні технології надають логістичним компаніям гнучкість і масштабованість. Вони забезпечують доступ до даних у будь-який час і з будь-якого місця, що дозволяє швидко адаптуватися до змінних умов ринку. Хмарні рішення також знижують витрати на інфраструктуру та обслуговування, забезпечуючи надійність і безперервність бізнес-процесів.

Аналітика великих даних стає все більш важливою для стратегічного управління логістикою. Вона дозволяє отримувати інсайти про ефективність операцій, виявляти можливості для оптимізації та підвищення продуктивності. Використання передових аналітичних інструментів допомагає компаніям швидко реагувати на зміни та забезпечувати високу якість обслуговування клієнтів.

Отже, перспективи розвитку логістичних інформаційних систем у майбутньому є надзвичайно обнадійливими. Інтеграція новітніх технологій, автоматизація, персоналізація послуг, екологічна стійкість, підвищення кібербезпеки, розвиток хмарних технологій та зростання ролі великих даних і

аналітики забезпечать подальше підвищення ефективності та конкурентоспроможності логістичних компаній. Ці інновації дозволять швидко адаптуватися до змінних ринкових умов та забезпечувати високу якість обслуговування клієнтів, що є ключовим фактором успіху в сучасному світі.

2. ВИМОГИ ЛОГІСТИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1. Аналіз потреб користувачів та стейкхолдерів

Успішне впровадження логістичних інформаційних систем (ЛІС) вимагає глибокого розуміння потреб користувачів та стейкхолдерів. Це дозволяє створити систему, яка відповідає їхнім очікуванням та вимогам, забезпечуючи максимальну ефективність та задоволеність. У логістиці користувачами ЛІС можуть бути менеджери з логістики, складські працівники, водії, клієнти та керівництво компанії. Кожна з цих груп має свої специфічні потреби та очікування від системи.

Менеджери з логістики потребують інструментів для планування та оптимізації маршрутів, управління запасами, моніторингу виконання замовлень та отримання детальних звітів і аналітики для прийняття обґрунтованих рішень. Вони повинні мати можливість швидко отримувати доступ до інформації про поточний стан логістичних операцій та прогнозувати майбутні потреби.

Складські працівники потребують інтуїтивно зрозумілих інтерфейсів для швидкого введення даних, управління складськими операціями та відстеження товарів. Важливим є забезпечення зручності використання системи, оскільки це дозволяє знизити час на навчання персоналу та мінімізувати кількість помилок при виконанні складських операцій.

Водії очікують на точні інструкції щодо маршрутів, доступ до інформації про вантажі та можливість швидко повідомляти про проблеми або затримки. Для них важливо мати мобільний доступ до системи, що дозволяє отримувати актуальну інформацію під час виконання доставки.

Клієнти, які є кінцевими користувачами логістичних послуг, хочуть мати змогу відстежувати свої замовлення в режимі реального часу, отримувати сповіщення про статус доставки та мати доступ до служби підтримки. Вони цінують швидку та надійну доставку, а також можливість легко зв'язатися з компанією у разі виникнення питань або проблем.

Керівництво компанії потребує високорівневих аналітичних інструментів для моніторингу ефективності логістичних операцій, виявлення проблем та

можливостей для покращення. Вони повинні мати доступ до ключових показників ефективності (KPI), які дозволяють оцінювати продуктивність та приймати стратегічні рішення.

Для проведення аналізу потреб користувачів та стейкхолдерів можна використовувати кілька методів, таких як опитування, інтерв'ю, спостереження та аналіз існуючих процесів. Опитування та інтерв'ю дозволяють зібрати детальну інформацію про очікування та проблеми користувачів, тоді як спостереження за роботою користувачів дає змогу виявити неявні потреби, які вони можуть не усвідомлювати.

Зібрані дані необхідно ретельно аналізувати, щоб зрозуміти, які функції та можливості повинна мати логістична інформаційна система. Наприклад, якщо багато складських працівників стикаються з проблемами при введенні даних, це може свідчити про необхідність спрощення інтерфейсу користувача. Якщо менеджери з логістики відчувають труднощі з отриманням актуальної інформації, система повинна забезпечувати швидкий доступ до даних та можливість генерування звітів у режимі реального часу.

На основі аналізу потреб визначаються конкретні вимоги до системи. Це можуть бути як функціональні вимоги (конкретні функції, які повинна виконувати система), так і нефункціональні (вимоги до продуктивності, безпеки, зручності використання тощо). Важливо забезпечити, щоб всі ключові потреби були враховані при розробці та впровадженні системи.

Постійна комунікація зі стейкхолдерами є критично важливою на всіх етапах розробки та впровадження системи. Це дозволяє своєчасно виявляти та вирішувати проблеми, а також забезпечує підтримку та зацікавленість усіх учасників процесу. Регулярні зустрічі, демонстрації прототипів та отримання зворотного зв'язку допомагають підтримувати проект у відповідності до очікувань користувачів та стейкхолдерів.

Врахування потреб користувачів та стейкхолдерів на ранніх етапах розробки логістичної інформаційної системи дозволяє створити продукт, який буде відповідати їхнім вимогам та сприяти підвищенню ефективності логістичних

операцій. Це, в свою чергу, забезпечує конкурентні переваги компанії на ринку та підвищує задоволеність клієнтів.

2.2. Функціональні вимоги до системи

Функціональні вимоги визначають конкретні функції, які повинна виконувати логістична інформаційна система (ЛІС). Вони охоплюють ключові операції, необхідні для ефективного управління логістичними процесами, такі як управління запасами, планування маршрутів, обробка замовлень, відстеження вантажів та інші важливі аспекти.

Однією з основних функцій ЛІС є управління запасами. Ця функція включає в себе:

Моніторинг рівнів запасів: ЛІС повинна надавати точну інформацію про поточні запаси на складах, включаючи кількість, місцезнаходження та стан товарів. Це допомагає уникнути дефіциту або надлишків товарів, що забезпечує ефективне управління ресурсами.

Автоматичне поповнення: Система повинна мати можливість автоматично генерувати замовлення на поповнення запасів на основі встановлених порогових значень та прогнозів попиту. Це знижує ризик дефіциту товарів і забезпечує безперервність логістичних процесів.

Управління складськими операціями: ЛІС повинна підтримувати операції з прийому, зберігання, переміщення та відвантаження товарів. Це включає автоматизацію процесів розміщення товарів на складі, управління складськими приміщеннями та оптимізацію використання простору.

Облік товарів: Система повинна вести точний облік товарів, включаючи партії, серійні номери та терміни придатності. Це особливо важливо для продуктів харчування та медикаментів, де необхідно контролювати умови зберігання та терміни придатності.

Ефективне планування маршрутів є критично важливим для зниження витрат на транспортування та підвищення швидкості доставки. Функціональні вимоги для планування маршрутів включають:

Оптимізація маршрутів: ЛІС повинна мати можливість автоматично планувати оптимальні маршрути доставки з урахуванням відстані, часу, витрат на паливо, умов дорожнього руху та інших факторів. Це допомагає знизити витрати на транспортування та забезпечити своєчасну доставку товарів.

Гнучкість у плануванні: Система повинна дозволяти оперативно вносити зміни до маршрутів у разі непередбачуваних обставин, таких як затори, поломки транспортних засобів або зміни в пріоритетах доставки. Це забезпечує гнучкість та адаптивність логістичних процесів.

Інструкції для водіїв: ЛІС повинна надавати водіям детальні інструкції щодо маршрутів, включаючи адреси, контактні дані отримувачів та особливі умови доставки. Це допомагає знизити кількість помилок та забезпечити високу точність виконання замовлень.

ЛІС повинна підтримувати повний цикл обробки замовлень, від їх прийому до доставки кінцевому споживачу. Функціональні вимоги включають:

Прийом замовлень: Система повинна забезпечувати можливість прийому замовлень через різні канали, такі як веб-сайти, мобільні додатки, електронну пошту та телефони. Це дозволяє клієнтам вибирати найбільш зручний спосіб замовлення товарів.

Підтвердження замовлень: ЛІС повинна автоматично підтверджувати отримання замовлень, надсилаючи клієнтам відповідні сповіщення. Це забезпечує прозорість процесу та підвищує довіру клієнтів до компанії.

Комплектація замовлень: Система повинна підтримувати процеси комплектації замовлень на складі, забезпечуючи точність та швидкість виконання. Це включає автоматизацію процесів збору товарів, пакування та підготовки до відправки.

Відвантаження: ЛІС повинна автоматизувати процеси відвантаження товарів, включаючи генерацію документів, пакування та маркування. Це допомагає знизити час на підготовку замовлень до відправки та забезпечує точність документального оформлення.

Прозорість та контроль над логістичними процесами є важливими аспектами для забезпечення високого рівня обслуговування клієнтів. Функціональні вимоги до відстеження вантажів включають:

Реальний час відстеження: ЛІС повинна забезпечувати можливість відстеження вантажів у режимі реального часу, надаючи точну інформацію про місцезнаходження товарів на кожному етапі доставки. Це підвищує прозорість та довіру клієнтів до логістичних послуг.

Сповіщення про статус: Система повинна автоматично надсилати сповіщення клієнтам про зміну статусу замовлення, такі як підтвердження відправлення, прибуття на проміжний склад або доставку до кінцевого пункту. Це забезпечує клієнтів актуальною інформацією про їхні замовлення.

Історія переміщень: ЛІС повинна зберігати історію переміщень вантажів, що дозволяє аналізувати логістичні операції та виявляти можливості для покращення. Це також важливо для забезпечення прозорості та контролю над логістичними процесами.

Забезпечення високого рівня обслуговування клієнтів є ключовим фактором успіху для логістичних компаній. Функціональні вимоги включають:

Підтримка клієнтів: ЛІС повинна забезпечувати інтеграцію з системами підтримки клієнтів, такими як CRM (Customer Relationship Management), для ефективного управління запитами та скаргами. Це допомагає забезпечити високу якість обслуговування та швидке реагування на запити клієнтів.

Персоналізація обслуговування: Система повинна дозволяти налаштовувати інтерфейси та послуги відповідно до індивідуальних потреб клієнтів. Це підвищує задоволеність клієнтів та забезпечує індивідуальний підхід до кожного клієнта.

Аналітика та звітність: ЛІС повинна надавати інструменти для аналізу даних про взаємодію з клієнтами, що дозволяє підвищувати якість обслуговування та виявляти тенденції в поведінці споживачів. Це допомагає компанії краще розуміти потреби клієнтів та адаптувати свої послуги відповідно до їхніх очікувань.

Таким чином, функціональні вимоги до логістичної інформаційної системи охоплюють широкий спектр завдань, необхідних для ефективного управління

логістичними процесами. Врахування цих вимог при розробці та впровадженні ЛІС дозволяє забезпечити високий рівень продуктивності, ефективності та обслуговування клієнтів.

2.3. Нефункціональні вимоги: безпека, продуктивність, масштабованість

Нефункціональні вимоги є основоположними для забезпечення стабільної та ефективної роботи логістичної інформаційної системи (ЛІС). Вони визначають, наскільки надійною, швидкодіючою та адаптивною буде система. Розглянемо основні аспекти цих вимог.

Безпека є першочерговим завданням для будь-якої інформаційної системи, особливо коли мова йде про логістику, де обробляється велика кількість конфіденційних даних. Це стосується не лише інформації про товари, але й персональних даних клієнтів та співробітників, фінансової інформації та інших критично важливих даних. Система повинна мати засоби шифрування даних як під час їх передачі, так і при зберіганні. Це гарантує, що навіть у випадку перехоплення дані залишаться незрозумілими для зловмисників.

Аутентифікація та авторизація користувачів є невід'ємною частиною безпеки. Користувачі повинні проходити перевірку особи, щоб отримати доступ до системи, а їх права доступу повинні бути чітко обмежені відповідно до ролей і завдань. Це запобігає несанкціонованому доступу до важливих функцій і даних. Журнали аудиту є ще одним важливим аспектом безпеки. Вони фіксують всі дії користувачів у системі, що дозволяє відстежувати підозрілу активність та проводити розслідування у випадку інцидентів. Крім того, система повинна підтримувати регулярне резервне копіювання даних, щоб у випадку збоїв або втрати інформації можна було швидко відновити всі важливі дані. Це забезпечує безперервність бізнес-процесів і мінімізує ризики втрати інформації.

Продуктивність логістичної інформаційної системи визначає, наскільки ефективно вона виконує свої функції і як швидко реагує на запити користувачів. Висока продуктивність системи є критично важливою для забезпечення оперативної роботи користувачів та підтримки високого рівня обслуговування

клієнтів. ЛІС повинна забезпечувати швидке завантаження інтерфейсів, миттєве оновлення даних та швидке виконання операцій. Це особливо важливо під час пікових навантажень, коли обсяг оброблюваних даних значно зростає. Система повинна бути здатною ефективно обробляти великі обсяги даних, зокрема обробку великої кількості замовлень, відстеження вантажів та генерацію звітів у реальному часі. Відмовостійкість системи також є важливим аспектом продуктивності. Це означає, що система повинна бути здатною продовжувати функціонувати навіть у випадку збоїв окремих компонентів. Наприклад, якщо один з серверів виходить з ладу, система повинна автоматично перенаправляти навантаження на інші сервери без переривання роботи користувачів.

Масштабованість визначає здатність системи адаптуватися до зростаючих обсягів даних, збільшення кількості користувачів та змін у бізнес-процесах. Це означає, що система повинна мати можливість розширювати свої ресурси без значних змін у архітектурі. Під горизонтальною масштабованістю розуміється можливість додавання нових серверів або компонентів для збільшення продуктивності. Це дозволяє системі рости разом із бізнесом, забезпечуючи стабільну роботу навіть при значному збільшенні навантаження. Вертикальна масштабованість означає можливість збільшення потужності існуючих серверів шляхом додавання додаткових ресурсів, таких як процесори та пам'ять. Це забезпечує гнучкість та можливість швидко реагувати на зростаючі вимоги бізнесу.

Гнучкість адаптації системи є також важливою частиною масштабованості. Це означає, що система повинна мати модульну архітектуру, яка дозволяє легко додавати нові функції та інтегруватися з іншими системами та сервісами. Така гнучкість дозволяє швидко адаптуватися до змін у бізнес-процесах та впроваджувати нові технології, що є критично важливим у сучасному динамічному середовищі.

Таким чином, безпека, продуктивність та масштабованість є ключовими нефункціональними вимогами для логістичних інформаційних систем. Вони забезпечують надійність, ефективність та гнучкість системи, що є важливими для успішного функціонування логістичних процесів. Врахування цих вимог при

розробці та впровадженні ЛІС дозволяє забезпечити високий рівень обслуговування клієнтів, ефективне управління ресурсами та підвищення конкурентоспроможності компанії.

2.4. Регуляторні вимоги та стандарти

Логістичні інформаційні системи (ЛІС) повинні відповідати широкому спектру регуляторних вимог та стандартів, щоб забезпечити законність, безпеку та ефективність своїх операцій. Дотримання цих вимог гарантує, що системи працюють у рамках встановлених норм та правил, забезпечуючи захист даних, екологічну безпеку та відповідність міжнародним стандартам.

Захист даних є одним з ключових аспектів, який регулюється різними законами та стандартами по всьому світу. Наприклад, у Європейському Союзі діє Загальний регламент про захист даних (GDPR), який встановлює суворі правила щодо збирання, зберігання та обробки персональних даних. Відповідно до GDPR, компанії зобов’язані забезпечувати конфіденційність та безпеку персональних даних своїх клієнтів і співробітників. Це включає впровадження відповідних технічних і організаційних заходів для захисту даних від несанкціонованого доступу, втрати або пошкодження. ЛІС повинні мати механізми для шифрування даних, управління доступом та ведення журналів аудиту, щоб відповідати вимогам GDPR. У США діє Закон про конфіденційність споживачів Каліфорнії (CCPA), який також встановлює вимоги до обробки персональних даних, схожі на GDPR.

Екологічні стандарти набувають все більшої важливості в логістиці. Багато країн запровадили регулювання викидів парникових газів та інших шкідливих речовин, що вимагає від логістичних компаній вживати заходів для зменшення свого екологічного впливу. ЛІС можуть допомогти оптимізувати маршрути доставки, зменшуючи витрати палива та викиди CO2. Це не лише допомагає дотримуватись екологічних стандартів, але й сприяє зниженню операційних витрат і покращенню репутації компанії. Наприклад, Європейський Союз має програму «Green Freight Europe», яка заохочує компанії до впровадження екологічних логістичних рішень.

Стандарти безпеки харчових продуктів є критично важливими для компаній, що займаються транспортуванням та зберіганням продуктів харчування. Наприклад, стандарти HACCP (Hazard Analysis and Critical Control Points) встановлюють вимоги до управління безпекою харчових продуктів, включаючи моніторинг умов зберігання, транспортування та обробки продуктів. ЛІС повинні мати можливість відстежувати температуру, вологість та інші умови зберігання, а також надавати звіти про відповідність цим стандартам. У США діє Закон про модернізацію безпеки харчових продуктів (FSMA), який вимагає від компаній забезпечувати простежуваність харчових продуктів через всю логістичну мережу.

Для перевезення небезпечних вантажів існують спеціальні регуляторні вимоги та стандарти. Наприклад, правила ADR (Accord Dangereux Routier) в Європі визначають вимоги до перевезення небезпечних матеріалів автотранспортом. ЛІС повинні підтримувати функції для забезпечення відповідності цим вимогам, включаючи управління документами, моніторинг умов транспортування та надання інформації про небезпечні вантажі. У США діє Федеральний закон про перевезення небезпечних матеріалів (Hazmat), який регулює транспортування небезпечних речовин і вимагає від компаній дотримання суворих правил безпеки.

Міжнародні стандарти управління якістю також важливі для логістичних інформаційних систем. Стандарти ISO (International Organization for Standardization) встановлюють вимоги до систем управління якістю, які можуть застосовуватися до логістичних процесів. Наприклад, стандарт ISO 9001 визначає вимоги до систем управління якістю, які можуть допомогти логістичним компаніям покращити свої процеси та забезпечити високу якість обслуговування клієнтів. Відповідність цим стандартам може підвищити довіру клієнтів та сприяти укладенню контрактів з міжнародними партнерами. Інші важливі стандарти включають ISO 28000, який визначає вимоги до систем управління безпекою ланцюга постачання, та ISO 14001, який стосується екологічного управління.

ЛІС також повинні враховувати регуляторні вимоги, пов'язані з митним оформленням та міжнародною торгівлею. Наприклад, митні правила визначають вимоги до документів, які повинні супроводжувати вантажі під час міжнародних

перевезень. Система повинна підтримувати автоматизацію підготовки та подання митних декларацій, що знижує ризик помилок та затримок при проходженні митного контролю. У цьому контексті важливими є такі міжнародні угоди, як Конвенція про спрощення митних процедур та Гармонізована система опису та кодування товарів.

Дотримання стандартів інформаційної безпеки є ще одним важливим аспектом для ЛІС. Стандарти ISO/IEC 27001 визначають вимоги до систем управління інформаційною безпекою, що включають заходи для захисту інформації від загроз, таких як кібератаки та втрати даних. Відповідність цим стандартам допомагає забезпечити надійний захист даних та підтримувати довіру клієнтів і партнерів. Важливими аспектами є шифрування даних, контроль доступу, безпека мереж та управління інцидентами.

Таким чином, регуляторні вимоги та стандарти відіграють ключову роль у забезпеченні законності, безпеки та ефективності логістичних інформаційних систем. Відповідність цим вимогам гарантує, що система працює у рамках встановлених норм, забезпечуючи захист даних, екологічну безпеку та відповідність міжнародним стандартам. Це сприяє підвищенню довіри клієнтів, зниженню ризиків та забезпеченню стабільної роботи логістичних процесів. Врахування цих вимог при розробці та впровадженні ЛІС є критично важливим для успішного функціонування логістичних компаній у сучасному глобалізованому світі.

2.5. Основні архітектурні шаблони інформаційних систем

Архітектурні шаблони визначають структуру та взаємодію компонентів інформаційної системи, включаючи логістичні інформаційні системи (ЛІС). Вибір архітектурного шаблону впливає на продуктивність, масштабованість, гнучкість і надійність системи. Розглянемо основні архітектурні шаблони, їхні переваги, недоліки та приклади застосування.

Клієнт-серверна архітектура є одним з найбільш поширених архітектурних шаблонів. У цій архітектурі система поділяється на клієнтську та серверну частини. Клієнтська частина надає інтерфейс користувача та взаємодіє з користувачами, тоді

як серверна частина відповідає за обробку даних, виконання бізнес-логіки та управління базами даних.

Основні переваги клієнт-серверної архітектури включають розподіл навантаження між клієнтом та сервером, централізоване управління даними та можливість масштабування сервера для підвищення продуктивності. Проте є й недоліки, такі як вразливість до збоїв сервера, складність адміністрування та потреба в надійних мережевих з'єднаннях.

Прикладом клієнт-серверної архітектури є система управління запасами, де клієнтські додатки використовуються для введення даних про товари, а сервер обробляє ці дані, зберігаючи їх у базі даних і забезпечуючи аналітичні звіти.

Мікросервісна архітектура стає все більш популярною завдяки своїй гнучкості та можливостям масштабування. У цій архітектурі система складається з набору незалежних сервісів, кожен з яких виконує певну функцію. Мікросервіси можуть розгортатися, оновлюватися та масштабуватися незалежно один від одного, що забезпечує високу гнучкість і адаптивність.

Основні переваги мікросервісної архітектури включають можливість незалежного розвитку та розгортання окремих компонентів, спрощення тестування та відновлення після збоїв, а також легкість інтеграції нових технологій. Проте є і виклики, такі як складність управління міжсервісною комунікацією, потреба в розвиненій системі оркестрації та моніторингу.

Прикладом використання мікросервісної архітектури є логістична платформа, де окремі сервіси відповідають за управління запасами, планування маршрутів, обробку замовлень та моніторинг доставки. Кожен сервіс може масштабуватися залежно від навантаження, що забезпечує високу продуктивність системи в цілому.

Одноріневa архітектура (Monolithic Architecture) є традиційним підходом, де вся система розгортається як єдине ціле. Цей підхід може бути ефективним для невеликих систем або проектів з чітко визначеними вимогами, де зміни рідкісні.

Перевагами одноріневої архітектури є простота розробки та розгортання, легкість тестування та відсутність складнощів з інтеграцією компонентів. Недоліки

включають складність масштабування, важкість впровадження змін та ризик впливу змін на всю систему.

Прикладом однорівневої архітектури є невелика логістична компанія, яка використовує єдину програму для управління замовленнями, запасами та доставкою. Всі функції системи інтегровані в один додаток, що спрощує його підтримку, але ускладнює масштабування при зростанні обсягів бізнесу.

Сервіс-орієнтована архітектура (SOA) базується на використанні сервісів для виконання бізнес-функцій. Сервіси в SOA можуть бути незалежними компонентами, що взаємодіють через стандартизовані протоколи.

Перевагами SOA є можливість повторного використання сервісів, гнучкість у адаптації до змін бізнес-вимог та легкість інтеграції з іншими системами. Проте цей підхід може бути складним у реалізації через потребу в управлінні сервісами та забезпеченні їх сумісності.

Прикладом SOA є велика логістична компанія, яка використовує сервіси для управління різними аспектами своєї діяльності, такими як обробка замовлень, управління запасами, планування маршрутів та моніторинг доставки. Сервіси можуть взаємодіяти між собою та з іншими системами через стандартизовані інтерфейси, що забезпечує гнучкість і масштабованість.

Архітектура з використанням подій (Event-Driven Architecture) фокусується на асинхронній обробці подій, де компоненти системи реагують на події, що відбуваються у реальному часі. Цей підхід дозволяє створювати високомасштабовані та реактивні системи.

Основні переваги архітектури, заснованої на подіях, включають високу масштабованість, гнучкість у обробці даних у реальному часі та зменшення залежностей між компонентами. Недоліками можуть бути складність управління подіями та потреба в надійних механізмах обробки помилок.

Прикладом архітектури з використанням подій є система відстеження вантажів у режимі реального часу, де кожна зміна статусу вантажу (наприклад, відправлення, прибуття на склад, доставка) генерує подію, на яку реагують відповідні компоненти системи.

Шарова архітектура (Layered Architecture) є одним з класичних архітектурних шаблонів, який передбачає поділ системи на декілька шарів, кожен з яких відповідає за певний аспект функціональності. Зазвичай виділяють наступні шари: презентаційний шар (інтерфейс користувача), логічний шар (бізнес-логіка), шар доступу до даних (робота з базами даних) та шар зберігання даних (бази даних).

Перевагами шарової архітектури є модульність, що дозволяє легко оновлювати або замінювати окремі шари без впливу на інші, спрощення розробки та тестування, а також полегшення повторного використання компонентів. Недоліками можуть бути зниження продуктивності через додаткові витрати на обробку запитів між шарами та складність управління залежностями між шарами.

Прикладом шарової архітектури є логістична система, де презентаційний шар відповідає за інтерфейс користувача (веб-додаток або мобільний додаток), логічний шар виконує обробку замовлень, планування маршрутів та управління запасами, шар доступу до даних забезпечує взаємодію з базами даних, а шар зберігання даних відповідає за зберігання інформації про товари, замовлення та клієнтів.

Архітектура на основі контейнерів (Container-Based Architecture) стає все більш популярною завдяки своїй здатності ізолювати окремі компоненти та забезпечувати легкість розгортання. Контейнери дозволяють упаковувати мікросервіси або інші компоненти разом з усіма їхніми залежностями, що спрощує їх перенесення та масштабування.

Перевагами контейнерної архітектури є портативність, легкість розгортання та масштабування, а також ізоляція додатків, що знижує ризик конфліктів між ними. Недоліки включають потребу в управлінні контейнерами та оркестрації, а також складність забезпечення безпеки та моніторингу.

Прикладом контейнерної архітектури є велика логістична платформа, де кожен мікросервіс запускається в окремому контейнері. Це дозволяє легко масштабувати кожен сервіс залежно від навантаження та забезпечує незалежне оновлення та розгортання.

Узагальнюючи, вибір архітектурного шаблону для логістичної інформаційної системи залежить від конкретних потреб та вимог бізнесу, масштабованості, продуктивності, гнучкості та інших факторів. Кожен з описаних шаблонів має свої переваги та недоліки, і правильний вибір залежить від специфіки проекту та цілей, які ставляться перед системою. Врахування цих аспектів дозволяє забезпечити ефективне функціонування логістичних процесів, високу надійність та гнучкість системи у відповідності до змінних умов ринку.

2.6. Клієнт-серверна архітектура: переваги та недоліки

Клієнт-серверна архітектура є однією з найпоширеніших моделей побудови інформаційних систем, зокрема логістичних інформаційних систем (ЛІС). Вона передбачає розподіл системи на клієнтську частину, що взаємодіє з користувачем, та серверну частину, що обробляє запити та управляє даними. Такий підхід має свої переваги та недоліки, які важливо враховувати при розробці та впровадженні системи.

Однією з ключових переваг клієнт-серверної архітектури є можливість розподілу обчислювальних завдань між клієнтськими і серверними компонентами. Клієнтська частина забезпечує інтерфейс користувача і виконує завдання, пов'язані з його взаємодією із системою, тоді як серверна частина обробляє складні обчислення, зберігає та керує даними. Це підвищує ефективність використання ресурсів і покращує загальну продуктивність системи. Наприклад, у системах електронної комерції, таких як Amazon, клієнтська частина забезпечує зручний інтерфейс для перегляду товарів, оформлення замовлень та оплати, тоді як серверна частина обробляє всі запити, керує базами даних, стежить за наявністю товарів та управляє логістичними процесами.

Серверна частина системи зберігає всі дані, що забезпечує централізоване управління інформацією. Це полегшує процеси резервного копіювання, відновлення даних та забезпечення їхньої узгодженості. У банківських системах, таких як ті, що використовуються в JPMorgan Chase, клієнтські додатки дозволяють користувачам здійснювати операції, керувати рахунками та отримувати

інформацію про свої фінанси, тоді як серверна частина обробляє запити, зберігає дані про рахунки та транзакції, забезпечуючи їхню безпеку.

Ще однією значною перевагою клієнт-серверної архітектури є її масштабованість. Додавання нових клієнтів або збільшення обсягу оброблюваних даних не потребує значних змін у серверній частині. Для підвищення продуктивності можна додавати нові сервери або збільшувати їхні ресурси. У системах управління підприємствами (ERP), таких як SAP, можливість масштабування дозволяє компаніям розширювати свої системи без значних перебудов, додаючи нові модулі та збільшуючи кількість користувачів.

Централізоване управління даними також забезпечує високий рівень безпеки. Адміністратори можуть контролювати доступ до даних, застосовувати політики безпеки та проводити аудит дій користувачів. У медичних інформаційних системах, таких як Epic Systems, централізоване зберігання медичних записів на серверах забезпечує високий рівень безпеки даних, їх шифрування та контроль доступу відповідно до стандартів HIPAA.

Клієнт-серверні системи підтримують віддалений доступ, що дозволяє користувачам працювати з системою з будь-якої точки світу. Це особливо важливо для логістичних систем, де працівники можуть знаходитись у різних географічних локаціях. У системах дистанційного навчання, таких як Moodle, студенти можуть взаємодіяти з навчальними матеріалами та виконувати завдання через інтернет, тоді як серверна частина обробляє запити та зберігає всі дані.

Проте клієнт-серверна архітектура має і свої недоліки. Центральна роль сервера робить систему вразливою до збоїв. Якщо сервер виходить з ладу, вся система може стати недоступною, що серйозно впливає на бізнес-процеси. У разі збою серверів компанії Netflix користувачі можуть втратити доступ до сервісу, що вимагає швидкого відновлення для мінімізації простоїв. Адміністрування клієнт-серверної системи також може бути складним, особливо у великих організаціях з численними користувачами та серверами. Це включає моніторинг продуктивності, оновлення програмного забезпечення та забезпечення безпеки. У великих

корпораціях, таких як Google, адміністрування серверів вимагає значних зусиль для забезпечення безперебійної роботи та безпеки даних.

Крім того, клієнт-серверна архітектура вимагає надійної та швидкої мережевої інфраструктури. Низька пропускна здатність мережі або висока затримка можуть негативно вплинути на продуктивність системи та зручність роботи користувачів. Це особливо актуально для віддалених офісів або мобільних користувачів. Впровадження та підтримка клієнт-серверної архітектури можуть бути дороговартісними, особливо для малих і середніх підприємств. Необхідно інвестувати в серверне обладнання, мережеву інфраструктуру, програмне забезпечення та послуги з адміністрування.

Узагальнюючи, клієнт-серверна архітектура має багато переваг, таких як розподіл навантаження, централізоване управління даними, масштабованість, безпека та можливість віддаленого доступу. Однак вона також має свої недоліки, зокрема вразливість до збоїв сервера, складність адміністрування, вимоги до мережевої інфраструктури, високі витрати на інфраструктуру та залежність від централізованої обробки. Вибір цієї архітектури повинен базуватися на аналізі потреб бізнесу, наявних ресурсів та вимог до продуктивності та надійності системи.

2.7. Розподілені системи: мікросервіси та їх застосування у логістиці

Розподілені системи та мікросервісна архітектура набувають все більшої популярності в сучасних IT-інфраструктурах, зокрема у логістиці. Цей підхід передбачає розділення системи на невеликі, незалежні сервіси, кожен з яких виконує окрему функцію та може розгортатися, оновлюватися і масштабуватися незалежно від інших. Така архітектура має свої переваги та недоліки, які важливо розглянути в контексті логістичних інформаційних систем.

Мікросервісна архітектура дозволяє підвищити гнучкість та адаптивність системи. Кожен мікросервіс виконує конкретну функцію, наприклад, управління запасами, планування маршрутів, обробка замовлень чи моніторинг доставки. Це забезпечує можливість незалежного розвитку, тестування та розгортання окремих компонентів системи. У логістичній платформі, де кожен мікросервіс відповідає за свій набір завдань, такі як управління запасами чи обробка замовлень, кожен сервіс

може масштабуватися окремо, що забезпечує високу продуктивність системи в цілому.

Одна з основних переваг мікросервісної архітектури полягає у можливості незалежного масштабування. У традиційній монолітній системі масштабування всього додатка може бути складним і дороговартісним, тоді як у мікросервісній архітектурі можна збільшувати ресурси тільки для тих компонентів, які цього потребують. Це дозволяє оптимально використовувати ресурси та забезпечувати високу продуктивність навіть при значному зростанні навантаження. Крім того, кожен мікросервіс може використовувати різні технології та платформи, що дозволяє розробникам вибирати найбільш підходящі інструменти для вирішення конкретних завдань. Наприклад, для обробки великих обсягів даних можна використовувати одну технологію, а для швидкої обробки транзакцій - іншу.

Ще однією важливою перевагою є підвищена надійність та відмовостійкість. Оскільки мікросервіси є незалежними компонентами, збій одного сервісу не призведе до зупинки всієї системи. Це знижує ризик простоїв та забезпечує безперервність бізнес-процесів. Наприклад, якщо сервіс, відповідальний за обробку замовлень, виходить з ладу, інші сервіси, такі як управління запасами чи моніторинг доставки, можуть продовжувати працювати без перешкод. Така архітектура дозволяє швидко виявляти та виправляти помилки без значних втрат для системи в цілому.

Мікросервісна архітектура також сприяє спрощенню процесів розробки та впровадження нових функцій. Оскільки кожен мікросервіс є незалежним, розробники можуть працювати над окремими частинами системи паралельно, що прискорює розробку та тестування. Це особливо важливо в умовах швидких змін ринкових вимог та необхідності оперативного впровадження нових функцій. Наприклад, компанія може швидко запровадити нову функцію для клієнтів, не чекаючи на оновлення всієї системи.

Однак, як і будь-яка архітектура, мікросервісна модель має свої недоліки. Управління великою кількістю мікросервісів може бути складним, особливо в масштабних системах. Потрібно забезпечити ефективну комунікацію між

сервісами, що може вимагати використання складних механізмів оркестрації та моніторингу. Для цього часто використовуються спеціалізовані платформи, такі як Kubernetes, які дозволяють автоматизувати розгортання, управління та масштабування контейнеризованих додатків. Використання таких платформ додає складності до адміністрування, але значно підвищує гнучкість та масштабованість системи.

Ще одним викликом є забезпечення безпеки в розподіленій системі. Кожен мікросервіс може мати свої вразливості, і забезпечення захисту всіх компонентів вимагає комплексного підходу до безпеки. Необхідно впроваджувати механізми аутентифікації та авторизації, шифрування даних та моніторингу безпеки на всіх рівнях системи. Наприклад, потрібно забезпечити, щоб кожен запит між мікросервісами був безпечним і аутентифікованим, щоб уникнути можливості несанкціонованого доступу.

У контексті логістики мікросервісна архітектура має численні приклади успішного застосування. Наприклад, компанія Uber використовує мікросервісну архітектуру для управління своєю глобальною логістичною мережею. Кожен мікросервіс відповідає за окремі аспекти роботи платформи, такі як замовлення поїздок, розрахунок тарифів, управління водіями та клієнтами, забезпечення безпеки та підтримка платіжних систем. Це дозволяє Uber швидко адаптуватися до змін на ринку, впроваджувати нові функції та масштабувати систему у відповідь на зростаючий попит. Завдяки цьому, компанія може швидко реагувати на змінні умови ринку, впроваджуючи нові послуги та покращуючи існуючі.

Інший приклад – компанія Amazon, яка також використовує мікросервіси для управління своїми складськими та логістичними процесами. Кожен мікросервіс відповідає за конкретні функції, такі як управління запасами, обробка замовлень, планування маршрутів доставки та моніторинг вантажів. Це забезпечує гнучкість, високу продуктивність та надійність системи, що є критично важливим для компанії з такими масштабами. Використання мікросервісів дозволяє Amazon підтримувати високу якість обслуговування клієнтів, швидко адаптуватися до змін попиту та забезпечувати ефективну роботу своїх логістичних процесів.

Загалом, мікросервісна архітектура є потужним інструментом для побудови розподілених систем, які можуть ефективно працювати у складних та динамічних умовах сучасного ринку. Вона дозволяє компаніям швидко реагувати на змінні вимоги, забезпечувати високу продуктивність та надійність, а також впроваджувати нові функції з мінімальними витратами часу та ресурсів. Проте, для успішного застосування мікросервісної архітектури необхідно враховувати її специфічні виклики, такі як управління міжсервісною комунікацією та забезпечення безпеки, та використовувати відповідні інструменти та практики для їх вирішення.

Узагальнюючи, мікросервісна архітектура має багато переваг, зокрема незалежне масштабування, підвищену надійність, гнучкість у розробці та впровадженні нових функцій. Однак вона також має свої виклики, такі як складність управління, забезпечення безпеки та потреба в ефективних механізмах оркестрації та моніторингу. Успішне застосування мікросервісної архітектури у логістиці, як показують приклади Uber та Amazon, дозволяє компаніям підвищити ефективність своїх процесів, швидко адаптуватися до змін та забезпечувати високу якість обслуговування клієнтів.

2.8. Приклади реалізації логістичних інформаційних систем на основі клієнт-серверної архітектури

Клієнт-серверна архітектура залишається однією з найпопулярніших моделей для побудови логістичних інформаційних систем (ЛІС) завдяки своїй простоті, ефективності та можливості централізованого управління даними. Реальні приклади впровадження ЛІС на основі клієнт-серверної архітектури демонструють її здатність забезпечувати надійну та ефективну роботу логістичних процесів у різних секторах економіки.

Приклад 1: FedEx

Одним з найяскравіших прикладів застосування клієнт-серверної архітектури у логістиці є компанія FedEx, яка використовує цю архітектуру для своєї глобальної системи управління доставками. Клієнтська частина системи включає додатки, які дозволяють клієнтам відстежувати свої відправлення, розраховувати вартість

доставки та оформлювати замовлення через інтернет. Серверна частина обробляє ці запити, зберігає інформацію про відправлення та забезпечує зв'язок з іншими системами компанії, такими як управління складами та транспортними засобами.

Ця система дозволяє FedEx ефективно управляти величезним обсягом даних про мільйони відправлень щодня. Завдяки централізованому управлінню даними, компанія може швидко обробляти замовлення, відстежувати місцезнаходження вантажів у реальному часі та забезпечувати високий рівень обслуговування клієнтів. Крім того, система забезпечує можливість масштабування, що дозволяє компанії адаптуватися до зростаючого попиту на свої послуги.

Приклад 2: UPS

Компанія UPS також активно використовує клієнт-серверну архітектуру для своєї логістичної інформаційної системи. Клієнтська частина включає додатки для мобільних пристроїв і веб-інтерфейси, які надають клієнтам можливість відстежувати свої відправлення, керувати доставками та отримувати сповіщення про статус відправлень. Серверна частина системи обробляє ці запити, зберігає дані про відправлення, маршрути та клієнтів, а також інтегрується з іншими системами компанії.

Система UPS забезпечує високий рівень надійності та безпеки даних, що є критично важливим для логістичної компанії. Завдяки централізованому управлінню даними та ефективній обробці запитів, UPS може швидко реагувати на зміни у вимогах клієнтів та забезпечувати безперебійну роботу своїх логістичних процесів. Масштабованість системи дозволяє компанії ефективно управляти зростаючими обсягами даних та адаптуватися до змін у бізнес-середовищі.

Приклад 3: DHL

DHL, одна з провідних логістичних компаній у світі, також використовує клієнт-серверну архітектуру для управління своїми логістичними операціями. Клієнтські додатки надають клієнтам доступ до інформації про відправлення, дозволяють планувати доставки та отримувати сповіщення про зміни у статусі відправлень. Серверна частина системи відповідає за обробку запитів, управління

базами даних та інтеграцію з іншими системами компанії, такими як управління транспортними засобами та складами.

Система DHL дозволяє компанії ефективно управляти своїми логістичними операціями, забезпечуючи високий рівень обслуговування клієнтів та швидку обробку запитів. Централізоване управління даними дозволяє компанії отримувати точну та актуальну інформацію про всі етапи доставки, що сприяє прийняттю обґрунтованих рішень та підвищенню ефективності бізнес-процесів. Масштабованість системи дозволяє DHL адаптуватися до змін у попиті та забезпечувати безперебійну роботу навіть у періоди пікових навантажень.

Приклад 4: Amazon

Amazon є одним з лідерів у впровадженні інноваційних логістичних рішень, використовуючи клієнт-серверну архітектуру для управління своєю логістичною мережею. Клієнтська частина системи включає веб-інтерфейси та мобільні додатки, які дозволяють клієнтам здійснювати покупки, відстежувати замовлення та отримувати інформацію про статус доставки. Серверна частина обробляє замовлення, керує запасами, планує маршрути доставки та забезпечує інтеграцію з логістичними центрами компанії.

Система Amazon забезпечує високу продуктивність та надійність, що дозволяє компанії обробляти мільйони замовлень щодня. Централізоване управління даними дозволяє компанії отримувати точну інформацію про наявність товарів, планувати маршрути доставки та забезпечувати своєчасне виконання замовлень. Масштабованість системи дозволяє Amazon швидко адаптуватися до зростаючого попиту на свої послуги, забезпечуючи високий рівень обслуговування клієнтів.

Приклад 5: Maersk

Maersk, провідна компанія у сфері морських перевезень, використовує клієнт-серверну архітектуру для управління своєю глобальною логістичною мережею. Клієнтські додатки надають клієнтам можливість відстежувати свої відправлення, отримувати інформацію про тарифи та планувати перевезення.

Серверна частина системи обробляє ці запити, зберігає дані про відправлення, маршрути та тарифи, а також забезпечує інтеграцію з іншими системами компанії.

Система Maersk дозволяє компанії ефективно управляти своїми логістичними операціями, забезпечуючи високий рівень обслуговування клієнтів та швидку обробку запитів. Централізоване управління даними дозволяє компанії отримувати точну та актуальну інформацію про всі етапи перевезення, що сприяє прийняттю обґрунтованих рішень та підвищенню ефективності бізнес-процесів. Масштабованість системи дозволяє Maersk адаптуватися до змін у попиті та забезпечувати безперебійну роботу навіть у періоди пікових навантажень.

Узагальнюючи, клієнт-серверна архітектура є ефективною моделлю для побудови логістичних інформаційних систем, забезпечуючи централізоване управління даними, високу продуктивність, надійність та масштабованість. Реальні приклади впровадження ЛІС у таких компаніях, як FedEx, UPS, DHL, Amazon та Maersk, демонструють її здатність забезпечувати надійну та ефективну роботу логістичних процесів у різних секторах економіки.

3. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Проектування інформаційної системи є критично важливим етапом у розробці логістичної платформи, яка забезпечить ефективне управління доставками. У цьому розділі розглядаються вимоги до системи, її архітектура, проектування бази даних, прикладного програмного інтерфейсу (API) та інтерфейсу користувача.

3.1. Визначення вимог до системи

Основними вимогами до логістичної інформаційної системи є набір функціональних та нефункціональних вимог, які забезпечують її ефективну роботу. Детальний розгляд цих вимог включає конкретні приклади.

3.1.1 Функціональні вимоги

Система повинна забезпечувати можливість додавання, редагування, видалення та перегляду інформації про посилки: менеджер з логістики може додавати нові посилки до системи, вводячи дані про відправника, отримувача, тип доставки, опис посилки, вагу та адреси відправки і отримання. Це дозволить ефективно управляти всіма відправленнями та відстежувати їхній статус у реальному часі.

Реальне відстеження місцезнаходження посилок за допомогою GPS є критично важливим для сучасних логістичних систем: клієнт може в будь-який момент перевірити, де знаходиться його посилка, використовуючи мобільний додаток або веб-інтерфейс. Це підвищує рівень довіри клієнтів до компанії та забезпечує прозорість логістичних операцій.

Система повинна автоматично сповіщувати клієнтів про статус їхніх посилок: після того як посилка буде відправлена, клієнт отримає SMS або електронне повідомлення з інформацією про відправлення та очікуваний час доставки. Це дозволить клієнтам бути в курсі всіх змін у статусі їхніх замовлень.

Система повинна мати можливість автоматичного та ручного планування маршрутів доставки. Автоматичне планування може враховувати фактори, такі як відстань, трафік та пріоритети замовлень для оптимізації маршрутів: система може

автоматично пропонувати оптимальні маршрути для кур'єрів, зменшуючи час доставки та витрати на пальне.

Система повинна забезпечувати можливість створення, редагування, видалення та відстеження замовлень. Це включає всі етапи обробки замовлень, від їх створення до остаточної доставки: менеджери можуть створювати нові замовлення, редагувати існуючі, видаляти скасовані замовлення та відстежувати їхній статус на всіх етапах.

3.1.2 Нефункціональні вимоги

Система повинна обробляти не менше 1000 запитів на секунду. Це важливо для забезпечення швидкої та ефективної роботи навіть у періоди пікових навантажень: під час святкового сезону кількість замовлень може значно збільшуватися, і система повинна бути здатна обробляти великий обсяг даних без затримок.

Забезпечення захисту даних через шифрування та механізми аутентифікації є критично важливим для захисту конфіденційної інформації: дані про клієнтів, посилки та транзакції повинні бути захищені від несанкціонованого доступу. Використання сучасних методів шифрування та багатофакторної аутентифікації дозволяє забезпечити високий рівень безпеки.

Система повинна легко масштабуватися для підтримки зростаючої кількості користувачів. Це означає, що з ростом бізнесу і збільшенням кількості замовлень система повинна бути здатна адаптуватися без значних перебудов: при зростанні кількості користувачів в системі повинна бути закладена можливість додавати нові сервери для розподілу навантаження.

Забезпечення безперебійної роботи системи з мінімальним часом простою є важливим для підтримки високого рівня обслуговування клієнтів: система повинна мати резервні сервери та механізми автоматичного відновлення, щоб у разі збою одного з компонентів система могла продовжувати роботу без зупинки.

3.1.3 Приклади реалізації вимог

Розглянемо кілька прикладів реалізації цих вимог у реальних системах:

Amazon Logistics використовує складну систему для управління посилками, відстеження та планування маршрутів. Система забезпечує високий рівень продуктивності та безпеки, підтримуючи мільйони замовлень щодня.

DHL застосовує систему відстеження вантажів у реальному часі, що дозволяє клієнтам отримувати актуальну інформацію про свої посилки. Система також автоматично сповіщує клієнтів про зміни статусу їхніх замовлень.

FedEx використовує інтегровану систему для автоматичного планування маршрутів, що враховує різні фактори, такі як трафік та відстань, для оптимізації процесу доставки.

Визначення чітких вимог до системи та їх реалізація з використанням сучасних технологій та методів дозволяє створити ефективну та надійну логістичну інформаційну систему, яка задовольняє потреби як бізнесу, так і клієнтів.

3.2. Аналіз функціональних та нефункціональних вимог

Логістична інформаційна система повинна забезпечувати можливість додавання, редагування, видалення та перегляду інформації про посилки. В основі цієї функціональності лежить зручний інтерфейс для введення та обробки даних. Для реалізації цього функціоналу буде використано веб-технології на основі HTML, CSS, JavaScript (React.js) та серверну частину на Node.js. Дані про посилки зберігатимуться у базі даних PostgreSQL.

Система повинна надавати можливість реального відстеження місцезнаходження посилок за допомогою GPS. Для цього буде використано API, що інтегрується з GPS-трекерами, встановленими на транспортних засобах. Наприклад, кожна посилка отримує унікальний трекінговий номер, який дозволяє клієнтам відстежувати її шлях до кінцевого пункту. Реалізація включає використання технологій RESTful API на базі Node.js та інтеграцію з GPS-сервісами через HTTP-запити.

Система повинна автоматично сповіщувати клієнтів про статус їхніх посилок. Сповіщення можуть здійснюватися через SMS або електронну пошту. Для цього буде використано API для надсилання повідомлень (наприклад, Twilio для

SMS та SendGrid для електронної пошти). Серверна частина системи буде відповідальною за відправку повідомлень при зміні статусу посилки.

Система повинна підтримувати автоматичне та ручне планування маршрутів доставки. Автоматичне планування враховуватиме відстань, трафік та пріоритети замовлень для оптимізації маршрутів. Реалізація цієї функції вимагатиме використання алгоритмів оптимізації маршрутів, таких як алгоритм Дейкстри або алгоритм A*. Серверна частина буде написана на Node.js, а обробка даних буде здійснюватися за допомогою Google Maps API для отримання інформації про дорожню обстановку.

Система повинна забезпечувати можливість створення, редагування, видалення та відстеження замовлень. Для цього буде розроблено інтерфейс для менеджерів з логістики, що дозволить легко управляти замовленнями. Серверна частина буде відповідати за обробку запитів на створення та оновлення замовлень, взаємодіючи з базою даних PostgreSQL для збереження інформації.

Система повинна обробляти не менше 1000 запитів на секунду. Для цього необхідно забезпечити ефективну обробку запитів та масштабованість серверної частини. Використання технологій Node.js і кластеризації дозволить розподілити навантаження між декількома серверами.

Забезпечення захисту даних через шифрування та механізми аутентифікації є критично важливим. Всі дані передаватимуться через захищені канали (HTTPS), а для аутентифікації користувачів буде використовуватися JWT (JSON Web Tokens). База даних буде шифрувати конфіденційні дані, такі як особисті дані клієнтів.

Система повинна легко масштабуватися для підтримки зростаючої кількості користувачів. Використання Docker для контейнеризації та Kubernetes для оркестрації контейнерів дозволить легко масштабувати серверні компоненти системи у відповідь на зростання навантаження.

Забезпечення безперебійної роботи системи з мінімальним часом простою. Для досягнення цієї вимоги буде впроваджено механізми автоматичного відновлення та резервного копіювання даних. Система буде використовувати

розподілені бази даних з реплікацією даних для підвищення надійності та доступності.

3.3. Архітектура системи

3.3.1. Загальна структура системи

Система складається з наступних компонентів:

Клієнтська частина: веб-додаток і мобільний додаток, розроблені на основі React.js та React Native відповідно.

Серверна частина: сервери додатків на Node.js, які обробляють запити клієнтів та виконують бізнес-логіку.

База даних: PostgreSQL для зберігання інформації про посилки, клієнтів, маршрути та відстеження.

Інтеграція з API: використання сторонніх сервісів для GPS-відстеження (наприклад, Mapbox), SMS-сповіщень (Twilio) та електронної пошти (SendGrid).

3.3.2. Шарова архітектура сервера

Серверна частина має шарову архітектуру, що включає:

Презентаційний шар: обробка HTTP-запитів, аутентифікація та авторизація користувачів через JWT.

Логічний шар: бізнес-логіка для управління посилками, планування маршрутів, обробки замовлень та відстеження вантажів.

Шар доступу до даних: взаємодія з базою даних PostgreSQL, включаючи виконання SQL-запитів.

Шар інтеграції з API: взаємодія зі сторонніми сервісами для GPS-відстеження, сповіщень та інших функцій.

3.4. Проектування бази даних

3.4.1. Модель даних

Основним елементом системи є посилка (замовлення), яка містить такі поля: телефон отримувача, ПІБ отримувача, тип доставки, опис посилки, вага посилки, телефон відправника, ПІБ відправника, адреса відправки, адреса отримання, місто відправки, місто отримання, статус, дата створення замовлення, дата останнього оновлення статусу.

3.4.2. Вибір СУБД

Вибір системи управління базами даних (СУБД) є важливим етапом у проектуванні інформаційної системи, оскільки від цього залежить ефективність зберігання, обробки та доступу до даних. Для нашої логістичної інформаційної системи розглянемо кілька популярних СУБД, а також аргументацію вибору оптимального варіанту.

Вимоги до СУБД:

Продуктивність: СУБД повинна забезпечувати високу швидкість обробки запитів та підтримувати великий обсяг одночасних операцій.

Масштабованість: Система повинна легко масштабуватися для обробки зростаючого обсягу даних та кількості користувачів.

Надійність: СУБД повинна забезпечувати високу надійність та доступність даних, включаючи можливості резервного копіювання та відновлення.

Безпека: Система повинна підтримувати механізми захисту даних, такі як шифрування, контроль доступу та аудит.

Підтримка складних запитів: СУБД повинна ефективно обробляти складні SQL-запити та транзакції.

Вартість: Вартість ліцензії та обслуговування СУБД повинна бути прийнятною для проекту.

Огляд популярних СУБД:

PostgreSQL

Переваги:

- Висока продуктивність та масштабованість.
- Підтримка складних запитів та транзакцій.
- Високий рівень безпеки та надійності.
- Відкрите програмне забезпечення (безкоштовне).

Недоліки:

- Можливо, складніша конфігурація для великих обсягів даних порівняно з деякими комерційними СУБД.

MySQL

Переваги:

- Швидка та ефективна обробка запитів.
- Широко використовується, добре задокументована.
- Відкрите програмне забезпечення (безкоштовне).

Недоліки:

- Обмежені можливості для складних запитів порівняно з PostgreSQL.
- Менший набір вбудованих функцій безпеки.

Oracle Database

Переваги:

- Висока продуктивність та надійність.
- Широкий набір функцій для бізнес-аналітики та безпеки.
- Підтримка складних запитів та транзакцій.

Недоліки:

- Висока вартість ліцензії та обслуговування.
- Складність в налаштуванні та управлінні.

Microsoft SQL Server

Переваги:

- Висока продуктивність та інтеграція з іншими продуктами Microsoft.
- Широкий набір інструментів для адміністрування та безпеки.
- Підтримка складних запитів та транзакцій.

Недоліки:

- Висока вартість ліцензії.
- Залежність від екосистеми Microsoft.

Для нашої логістичної інформаційної системи обрано PostgreSQL з наступних причин:

Висока продуктивність та масштабованість: PostgreSQL забезпечує ефективну обробку великого обсягу запитів і дозволяє легко масштабувати систему у разі зростання обсягу даних та кількості користувачів.

Підтримка складних запитів та транзакцій: PostgreSQL має потужний механізм обробки SQL-запитів, що дозволяє ефективно управляти даними у складних логістичних процесах.

Надійність та безпека: PostgreSQL надає широкий набір функцій для забезпечення безпеки даних, включаючи шифрування, контроль доступу та аудит. Також система підтримує механізми резервного копіювання та відновлення даних.

Відкритий код та безкоштовне використання: PostgreSQL є відкритим програмним забезпеченням, що дозволяє знизити витрати на ліцензії та обслуговування, зберігаючи при цьому високу якість та функціональність.

Таким чином, вибір PostgreSQL як СУБД для нашої логістичної інформаційної системи забезпечує оптимальний баланс між продуктивністю, масштабованістю, надійністю та вартістю, що є критично важливим для успішного впровадження та експлуатації системи.

3.4.3. Створення схеми бази даних

Схема бази даних визначає структуру зберігання даних та їх взаємозв'язки, що є ключовим для ефективного функціонування логістичної інформаційної системи. У даному підрозділі ми розглянемо створення схеми бази даних на основі моделі даних, розробленої раніше.

Детальні лістинги коду наведені в додатку А

Створення схеми бази даних на основі розробленої моделі даних забезпечує ефективне зберігання та обробку інформації про посилки, маршрути та відстеження вантажів. Використання PostgreSQL як СУБД дозволяє реалізувати вимоги до продуктивності, масштабованості та надійності системи. Завдяки цьому система зможе забезпечити надійну та ефективну роботу у сфері логістики.

3.5. Проектування прикладного програмного інтерфейсу (API)

Прикладний програмний інтерфейс (API) є критичним компонентом логістичної інформаційної системи, що забезпечує взаємодію між клієнтською

частиною системи та серверною частиною. API дозволяє виконувати операції з даними, забезпечуючи доступ до функцій системи для зовнішніх додатків та користувачів.

3.5.1. Визначення основних функцій API

Основні функції API включають:

Управління посилками:

- Створення нових посилок.
- Редагування інформації про посилки.
- Видалення посилок.
- Перегляд інформації про посилки.

Відстеження посилок:

- Отримання інформації про поточне місцезнаходження та статус посилки.
- Додавання нових записів відстеження.

Управління клієнтами:

- Створення нових клієнтів.
- Редагування інформації про клієнтів.
- Видалення клієнтів.
- Перегляд інформації про клієнтів.

Планування маршрутів:

- Створення нових маршрутів.
- Перегляд інформації про маршрути.

3.5.2. Специфікація API

Специфікація API забезпечує чітке визначення операцій, які можуть бути виконані з даними, що зберігаються у системі, включаючи створення, редагування, видалення та перегляд інформації про посилки, клієнтів та маршрути. Це дозволяє забезпечити ефективну взаємодію між клієнтською та серверною частинами системи, а також з іншими зовнішніми сервісами.

Детальні лістинги коду наведені в додатку А

3.6. Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) є важливим етапом у розробці логістичної інформаційної системи. Від якості інтерфейсу залежить зручність використання системи, ефективність взаємодії користувачів з системою та загальний рівень задоволеності клієнтів.

3.6.1. Вимоги до інтерфейсу

Інтерфейс користувача повинен відповідати наступним вимогам:

Інтуїтивність: Інтерфейс повинен бути простим та зрозумілим для користувачів, навіть без попереднього досвіду роботи з подібними системами.

Доступність: Система повинна бути доступною з різних пристроїв, включаючи комп'ютери, планшети та смартфони.

Швидкодія: Інтерфейс повинен забезпечувати швидкий доступ до основних функцій системи без значних затримок.

Кроссплатформеність: Інтерфейс повинен однаково добре працювати на різних операційних системах та браузерах.

Візуальна привабливість: Інтерфейс повинен бути естетично привабливим, використовуючи сучасні принципи дизайну та кольорові схеми.

Локалізація: Система повинна підтримувати багатомовність, щоб забезпечити доступність для користувачів з різних країн.

Безпека: Інтерфейс повинен забезпечувати безпечний доступ до даних, включаючи механізми аутентифікації та авторизації.

3.6.2. Прототипи інтерфейсу

Для розробки та тестування інтерфейсу користувача були створені прототипи основних сторінок системи. Прототипи допомагають візуалізувати майбутній інтерфейс, перевірити зручність його використання та внести необхідні корективи на ранніх етапах розробки.

Головна сторінка: Містить огляд основних функцій системи, доступ до управління посилками, клієнтами та маршрутами.

Сторінка управління посилками: Включає форми для створення, редагування та видалення посилок, а також відображає список всіх посилок із можливістю пошуку та фільтрації.

Сторінка відстеження посилок: Містить інформацію про поточне місцезнаходження та статус посилки, включаючи історію змін статусу.

Сторінка управління клієнтами: Включає форми для створення, редагування та видалення клієнтів, а також відображає список всіх клієнтів із можливістю пошуку та фільтрації.

Сторінка планування маршрутів: Містить інструменти для автоматичного та ручного планування маршрутів доставки, відображає карту з маршрутами та ключовими точками.

Прототипи були створені за допомогою інструментів для дизайну інтерфейсів, таких як Figma або Adobe XD. Вони включають інтерактивні елементи, що дозволяють протестувати навігацію та функціональність інтерфейсу.

3.6.3. Вибір технологій для реалізації інтерфейсу

Для реалізації інтерфейсу користувача були обрані наступні технології:

React.js: Фреймворк для створення веб-додатків, який забезпечує високу продуктивність, гнучкість та зручність у розробці. React.js дозволяє створювати компоненти інтерфейсу, що легко повторно використовуються та підтримують реактивне оновлення даних.

React Native: Фреймворк для створення мобільних додатків, який дозволяє використовувати одну кодову базу для розробки додатків для iOS та Android. Це значно знижує витрати на розробку та підтримку мобільних додатків.

HTML5 та CSS3: Основні технології для створення структури та стилів веб-сторінок. Вони забезпечують підтримку сучасних стандартів веб-дизайну та дозволяють створювати адаптивні інтерфейси, що підходять для різних розмірів екранів.

Material-UI: Бібліотека компонентів для React.js, що реалізує дизайн-систему Material Design від Google. Використання Material-UI дозволяє створювати візуально привабливі та функціональні інтерфейси з мінімальними зусиллями.

Вибір цих технологій забезпечує створення сучасного, ефективного та зручного інтерфейсу користувача, який відповідає всім вимогам та дозволяє легко розширювати функціональність системи у майбутньому.

Детальні лістинги коду наведені в додатку А.

4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Вибір технологій та інструментів для реалізації

При виборі технологій та інструментів для реалізації логістичної інформаційної системи ми керувалися критеріями продуктивності, масштабованості, надійності та зручності в розробці. Основними мовами програмування стали JavaScript і TypeScript, які забезпечують ефективність як на клієнтській, так і на серверній частині. Використання TypeScript додало перевагу статичної типізації, що покращило якість коду і знизило ймовірність помилок.

Для серверної частини ми обрали Node.js. Цей серверний фреймворк відомий своєю продуктивністю завдяки асинхронній обробці запитів. Він забезпечує високу швидкодію і здатний обробляти велику кількість одночасних запитів, що є важливим для нашої системи.

На клієнтській частині ми використовували React.js для веб-додатка і React Native для мобільного додатка. React.js дозволяє створювати динамічні і інтерактивні інтерфейси, що робить користування додатком зручним і приємним. React Native, у свою чергу, забезпечує кросплатформенність, дозволяючи розробляти мобільні додатки для iOS та Android на основі єдиної кодової бази.

База даних обрана PostgreSQL. Ця реляційна СУБД відома своєю високою продуктивністю і надійністю. Вона підтримує складні SQL-запити і транзакції, що дозволяє ефективно обробляти великі обсяги даних і забезпечувати їхню цілісність.

Тестування системи є невід'ємною частиною розробки, тому ми обрали кілька інструментів для різних типів тестування. Jest і Mocha використовуються для юніт- та інтеграційного тестування, забезпечуючи перевірку окремих модулів і їх взаємодії. Selenium допомагає автоматизувати тестування веб-інтерфейсу, дозволяючи переконатися в коректній роботі додатка з точки зору користувача. Для тестування продуктивності обрали Apache JMeter, який дозволяє створювати навантаження і аналізувати швидкодію системи під різними умовами.

Для керування версіями коду використовували Git з хостингом на GitHub. Це дозволяє ефективно організувати командну роботу, відстежувати зміни в коді і

автоматизувати процеси CI/CD. Docker та Kubernetes стали невід'ємними інструментами для контейнеризації і оркестрації. Вони забезпечують ізоляцію середовищ виконання і легке масштабування додатка, що підвищує його надійність і гнучкість у розгортанні.

Інструменти Postman та Swagger використовувалися для тестування і документування API. Postman дозволяє створювати і виконувати HTTP-запити, що допомагає у налагодженні і тестуванні API. Swagger автоматично генерує документацію для API, роблячи процес інтеграції з зовнішніми сервісами зручнішим.

Вибір цих технологій і інструментів був обґрунтований необхідністю створення ефективної, масштабованої і надійної логістичної інформаційної системи, яка відповідає всім сучасним вимогам і забезпечує зручність в розробці, тестуванні і підтримці.

4.2. Реалізація серверної частини системи

Реалізація серверної частини логістичної інформаційної системи є ключовим етапом, що забезпечує обробку даних та взаємодію між клієнтськими додатками і базою даних. Важливою метою цього етапу є створення надійного та продуктивного серверного API, що підтримує всі необхідні функції для управління посилками, клієнтами та маршрутами.

Налаштування середовища розробки

Для початку реалізації серверної частини було встановлено Node.js, що дозволяє розробляти на JavaScript і TypeScript. Використання TypeScript забезпечує додаткову безпеку та якість коду завдяки статичній типізації.

Встановлені основні бібліотеки:

Express: фреймворк для створення веб-додатків і API.

Sequelize: ORM для роботи з PostgreSQL.

JWT (JSON Web Tokens): для аутентифікації та авторизації користувачів.

Реалізація основних маршрутів API

Створення основних маршрутів включає розробку CRUD (створення, читання, оновлення, видалення) операцій для кожного з основних об'єктів системи: посилок, клієнтів та маршрутів.

Інтеграція зі сторонніми сервісами

Для забезпечення можливості відстеження посилок у реальному часі ми інтегрували систему з GPS-трекерами. Використання сторонніх API дозволяє отримувати актуальні дані про місцезнаходження посилок і зберігати їх у нашій базі даних.

Приклад інтеграції з GPS-сервісом:

```
const axios = require('axios');

async function updateParcelLocation(parcelId) {
  const response = await axios.get(`https://api.gps-service.com/track/${parcelId}`);
  const locationData = response.data;
  await Parcel.update({ location: locationData.location }, { where: { id: parcelId } });
}
```

Аутентифікація та авторизація користувачів

Для забезпечення безпеки доступу до API ми використовували JWT (JSON Web Tokens). Користувачі аутентифікуються за допомогою токенів, що генеруються при вході в систему і передаються в заголовках запитів.

Реалізація серверної частини системи включає налаштування середовища, розробку основних маршрутів API, інтеграцію зі сторонніми сервісами для відстеження посилок та забезпечення аутентифікації користувачів. Використання сучасних технологій та фреймворків дозволяє створити продуктивну та надійну серверну частину, що відповідає всім вимогам логістичної інформаційної системи.

4.3. Реалізація клієнтської частини системи

Реалізація клієнтської частини логістичної інформаційної системи включає розробку веб-додатка та мобільного додатка, що забезпечують інтуїтивно зрозумілий інтерфейс для користувачів. Веб-додаток створено на основі React.js, а мобільний додаток — на основі React Native. Використання цих фреймворків

дозволяє забезпечити високу продуктивність, гнучкість та кроссплатформенність додатків.

Налаштування середовища розробки

Для початку роботи було створено базовий проект з використанням Create React App для веб-додатка та Create React Native App для мобільного додатка. Це забезпечило швидкий старт і готову структуру проекту.

Інтеграція з серверною частиною

Клієнтські додатки інтегровані з серверною частиною за допомогою RESTful API. Запити виконуються за допомогою бібліотеки Axios, що забезпечує зручний синтаксис для обробки HTTP-запитів.

Реалізація клієнтської частини системи включає налаштування середовища розробки, створення основних компонентів для управління посилками, інтеграцію з серверною частиною через RESTful API та забезпечення інтуїтивно зрозумілого інтерфейсу для користувачів. Використання сучасних технологій та фреймворків дозволяє створити ефективні та зручні у використанні веб та мобільні додатки.

Лістинги коду наведені в додатку В.

4.4. Тестування системи

Тестування системи є невід'ємною частиною розробки, що дозволяє забезпечити якість, надійність та безпеку логістичної інформаційної системи. У цьому розділі ми розглянемо різні види тестування, включаючи функціональне, продуктивне та безпекове тестування.

4.4.1. Види тестування

Тестування системи поділяється на кілька основних видів, кожен з яких спрямований на перевірку певних аспектів роботи системи:

Юніт-тестування: Перевірка окремих модулів або компонентів системи для забезпечення їхньої коректної роботи.

Інтеграційне тестування: Перевірка взаємодії між різними компонентами системи, щоб переконатися у їхній сумісності та правильності обміну даними.

Системне тестування: Комплексне тестування всієї системи для перевірки відповідності вимогам.

Продуктивне тестування: Перевірка системи на продуктивність та здатність обробляти велике навантаження.

Тестування безпеки: Перевірка системи на вразливості та забезпечення захищеності від можливих атак.

4.4.2. Тестування функціональності

Тестування функціональності включає юніт-тестування та інтеграційне тестування, спрямоване на перевірку коректної роботи функцій системи.

Юніт-тестування:

Для юніт-тестування використовуються фреймворки Jest і Mocha. Юніт-тести перевіряють окремі модулі системи, такі як функції для обробки запитів, маніпуляції даними та інші.

Приклад юніт-тесту:

```
import { addParcel } from './parcelService';

test('should add a new parcel', () => {
  const parcel = { receiver_name: 'John Doe', sender_name: 'Jane Smith', weight: 2.5
};
  const result = addParcel(parcel);
  expect(result).toHaveProperty('id');
  expect(result.receiver_name).toBe('John Doe');
});
```

Інтеграційні тести перевіряють взаємодію між різними компонентами системи, такими як клієнтська частина і серверна частина. Для інтеграційного тестування використовується Mocha.

Приклад інтеграційного тесту:

```
import request from 'supertest';
import app from '../app';

describe('Parcel API', () => {
  it('should create a new parcel', async () => {
    const response = await request(app)
      .post('/api/parcels')
      .send({ receiver_name: 'John Doe', sender_name: 'Jane Smith', weight: 2.5 });
    expect(response.status).toBe(201);
    expect(response.body).toHaveProperty('id');
  });
});
```

4.4.3. Тестування продуктивності

Тестування продуктивності допомагає оцінити здатність системи обробляти велику кількість запитів та забезпечувати стабільну роботу під навантаженням. Для цього використовується Apache JMeter.

Налаштування JMeter для тестування продуктивності:

Створення тестових сценаріїв: Сценарії включають різні типи запитів до API, такі як створення посилок, отримання списку посилок та оновлення інформації про посилки.

Запуск тестів: Запуск тестових сценаріїв з різним рівнем навантаження (наприклад, 100, 500 та 1000 одночасних користувачів).

Аналіз результатів: Аналіз часу відповіді, пропускної здатності та інших метрик для виявлення можливих вузьких місць у системі.

Приклад налаштування JMeter:

Створення тест-плану з кількома Thread Group, що моделюють різні сценарії використання системи.

Додавання HTTP Request для кожної операції API.

Конфігурація графічних звітів для аналізу результатів.

4.4.4. Тестування безпеки

Тестування безпеки спрямоване на виявлення вразливостей у системі та забезпечення її захищеності від атак. Для цього використовуються інструменти, такі як OWASP ZAP.

Процедура тестування безпеки:

Сканування вразливостей: Використання OWASP ZAP для автоматичного сканування веб-додатка на наявність поширених вразливостей, таких як SQL-ін'єкції, XSS (міжсайтовий скриптинг) та CSRF (міжсайтові запити підробки).

Аналіз результатів: Перевірка результатів сканування, визначення критичних вразливостей та їх усунення.

Тестування аутентифікації та авторизації: Перевірка механізмів аутентифікації (JWT) для захисту доступу до API, тестування контролю доступу до різних ресурсів.

Приклад налаштування OWASP ZAP:

Запуск автоматичного сканування цільового веб-додатка.

Аналіз звіту про вразливості та їхній вплив на систему.

Рекомендації щодо виправлення виявлених проблем.

4.5. Аналіз результатів тестування та оптимізація системи

Після проведення всіх видів тестування результати аналізуються для виявлення та виправлення проблем:

Аналіз результатів функціонального тестування: Перевірка відповідності системи функціональним вимогам. Усі виявлені помилки виправляються, після чого тести повторюються для підтвердження їх успішного проходження.

Аналіз результатів тестування продуктивності: Виявлення вузьких місць у системі, таких як повільні запити або недостатня пропускна здатність. Оптимізація коду та конфігурації серверів для підвищення продуктивності.

Аналіз результатів тестування безпеки: Виправлення виявлених вразливостей для забезпечення надійного захисту даних користувачів та стабільної роботи системи.

Оптимізація системи включає внесення змін до коду, налаштування інфраструктури та покращення процесів розгортання, що дозволяє забезпечити високу якість, продуктивність та безпеку логістичної інформаційної системи.

ВИСНОВКИ

У результаті проведеного дослідження були досягнуті наступні наукові та практичні результати, які підтвердили актуальність та ефективність запропонованих методів і технологій для розвитку логістичних інформаційних систем на основі клієнт-серверної архітектури.

Було проведено детальний аналіз існуючих ЛІС, визначено основні тенденції та виклики у сучасній логістиці, а також розглянуто роль інформаційних технологій у розвитку логістики. Було виявлено, що інтеграція новітніх технологій, таких як штучний інтелект, великі дані та Інтернет речей (IoT), є критично важливою для підвищення ефективності та адаптивності ЛІС.

На основі аналізу потреб користувачів та стейкхолдерів були сформульовані функціональні та нефункціональні вимоги до розроблюваної системи. Це включало вимоги до управління запасами, планування маршрутів, обробки замовлень, відстеження вантажів, забезпечення безпеки, продуктивності та масштабованості системи.

Розроблено архітектурний дизайн системи на основі клієнт-серверної моделі, яка забезпечує розподіл навантаження між клієнтськими та серверними компонентами, централізоване управління даними та можливість масштабування системи для підвищення продуктивності.

Крім того, було розроблено та реалізовано прототип логістичної інформаційної системи, яка включає функції управління запасами, планування маршрутів, обробки замовлень та відстеження вантажів. Система була створена з урахуванням вимог безпеки, продуктивності та масштабованості, що дозволило забезпечити її стабільну та ефективну роботу.

Проведено тестування прототипу системи на базі реальних даних логістичної компанії.

Таким чином, проведене дослідження підтвердило ефективність використання клієнт-серверної архітектури для розробки логістичних інформаційних систем. Запропоновані методи та технології дозволяють створити

масштабовану, гнучку та надійну систему, яка забезпечує високу продуктивність та безпеку, а також інтеграцію з новітніми технологіями. Результати дослідження мають практичне значення для логістичних компаній, сприяючи підвищенню ефективності їх діяльності та конкурентоспроможності на ринку.

ДОДАТОК А: ЛІСТИНГИ КОДУ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ

А.1. Реалізація маршрутів для посилок

ParcelRoutes.js:

```
const express = require('express');
const router = express.Router();
const { Parcel } = require('../models');

// Створення нової посилки
router.post('/parcels', async (req, res) => {
  const newParcel = await Parcel.create(req.body);
  res.status(201).json(newParcel);
});

// Отримання інформації про конкретну посилку
router.get('/parcels/:id', async (req, res) => {
  const parcel = await Parcel.findByPk(req.params.id);
  res.status(200).json(parcel);
});

// Оновлення інформації про посилку
router.put('/parcels/:id', async (req, res) => {
  const updatedParcel = await Parcel.update(req.body, { where: { id: req.params.id
} });
  res.status(200).json(updatedParcel);
});

// Видалення посилки
router.delete('/parcels/:id', async (req, res) => {
  await Parcel.destroy({ where: { id: req.params.id } });
  res.status(204).send();
});

module.exports = router;
```

А.2. Реалізація маршрутів для клієнтів

CustomerRoutes.js:

```
const express = require('express');
const router = express.Router();
const { Customer } = require('../models');

// Створення нового клієнта
router.post('/customers', async (req, res) => {
  const newCustomer = await Customer.create(req.body);
  res.status(201).json(newCustomer);
});

// Отримання інформації про конкретного клієнта
router.get('/customers/:id', async (req, res) => {
  const customer = await Customer.findByPk(req.params.id);
  res.status(200).json(customer);
});

// Оновлення інформації про клієнта
router.put('/customers/:id', async (req, res) => {
```

```
  const updatedCustomer = await Customer.update(req.body, { where: { id:
req.params.id } });
  res.status(200).json(updatedCustomer);
});

// Видалення клієнта
router.delete('/customers/:id', async (req, res) => {
  await Customer.destroy({ where: { id: req.params.id } });
  res.status(204).send();
});

module.exports = router;
```

А.3. Реалізація маршрутів для маршрутів

RouteRoutes.js:

```
const express = require('express');
const router = express.Router();
const { Route } = require('../models');

// Створення нового маршруту
router.post('/routes', async (req, res) => {
  const newRoute = await Route.create(req.body);
  res.status(201).json(newRoute);
});

// Отримання інформації про конкретний маршрут
router.get('/routes/:id', async (req, res) => {
  const route = await Route.findByPk(req.params.id);
  res.status(200).json(route);
});

module.exports = router;

А.4. Інтеграція з GPS-сервісом
GPSIntegration.js:
const axios = require('axios');

async function updateParcelLocation(parcelId) {
  const response = await axios.get(`https://api.gps-
service.com/track/${parcelId}`);
  const locationData = response.data;
  await Parcel.update({ location: locationData.location }, { where: { id: parcelId
} });
}

module.exports = { updateParcelLocation };
```

А.5. Аутентифікація користувачів

AuthRoutes.js:

```
const express = require('express');
const jwt = require('jsonwebtoken');
const { User } = require('../models');
const router = express.Router();

router.post('/login', async (req, res) => {
  const user = await User.findOne({ where: { email: req.body.email } });
  if (!user || !user.validPassword(req.body.password)) {
```

```
    return res.status(401).json({ message: 'Invalid credentials' });
  }
  const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expiresIn: '1h'
});
  res.status(200).json({ token });
});

module.exports = router;
```

ДОДАТОК В: ЛІСТИНГИ КОДУ ДЛЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

В.1. Компоненти для управління посилками (веб-додаток)

ParcelList.js:

```
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchParcels } from '../actions/parcelActions';
import ParcelItem from './ParcelItem';

const ParcelList = () => {
  const dispatch = useDispatch();
  const parcels = useSelector(state => state.parcels);

  useEffect(() => {
    dispatch(fetchParcels());
  }, [dispatch]);

  return (
    <div>
      <h1>Parcel List</h1>
      <ul>
        {parcels.map(parcel => (
          <ParcelItem key={parcel.id} parcel={parcel} />
        ))}
      </ul>
    </div>
  );
};

export default ParcelList;
```

ParcelForm.js:

```
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { createParcel } from '../actions/parcelActions';

const ParcelForm = () => {
  const [formData, setFormData] = useState({
    receiver_phone: '',
    receiver_name: '',
    delivery_type: '',
    description: '',
    weight: '',
    sender_phone: '',
    sender_name: '',
    sender_address: '',
    receiver_address: '',
    sender_city: '',
    receiver_city: '',
  });

  const dispatch = useDispatch();

  const handleChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.value
```

```
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    dispatch(createParcel(formData));
  };

  return (
    <form onSubmit={handleSubmit}>
      <input   type="text"   name="receiver_phone"   value={formData.receiver_phone}
onChange={handleChange} placeholder="Receiver Phone" required />
      <input   type="text"   name="receiver_name"   value={formData.receiver_name}
onChange={handleChange} placeholder="Receiver Name" required />
      <input   type="text"   name="delivery_type"   value={formData.delivery_type}
onChange={handleChange} placeholder="Delivery Type" required />
      <input    type="text"    name="description"    value={formData.description}
onChange={handleChange} placeholder="Description" required />
      <input       type="number"       name="weight"       value={formData.weight}
onChange={handleChange} placeholder="Weight" required />
      <input    type="text"    name="sender_phone"    value={formData.sender_phone}
onChange={handleChange} placeholder="Sender Phone" required />
      <input    type="text"    name="sender_name"    value={formData.sender_name}
onChange={handleChange} placeholder="Sender Name" required />
      <input   type="text"   name="sender_address"   value={formData.sender_address}
onChange={handleChange} placeholder="Sender Address" required />
      <input type="text" name="receiver_address" value={formData.receiver_address}
onChange={handleChange} placeholder="Receiver Address" required />
      <input    type="text"    name="sender_city"    value={formData.sender_city}
onChange={handleChange} placeholder="Sender City" required />
      <input   type="text"   name="receiver_city"   value={formData.receiver_city}
onChange={handleChange} placeholder="Receiver City" required />
      <button type="submit">Submit</button>
    </form>
  );
};

export default ParcelForm;
```

В.2. Компоненти для управління посилками (мобільний додаток)

ParcelListScreen.js:

```
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchParcels } from '../actions/parcelActions';
import { FlatList, Text, View } from 'react-native';

const ParcelListScreen = () => {
  const dispatch = useDispatch();
  const parcels = useSelector(state => state.parcels);

  useEffect(() => {
    dispatch(fetchParcels());
  }, [dispatch]);

  return (
    <View>
      <Text>Parcel List</Text>
      <FlatList
        data={parcels}
        keyExtractor={item => item.id.toString()}
        renderItem={({ item }) => <Text>{item.receiver_name}</Text>}
```

```
      />
    </View>
  );
};

export default ParcelListScreen;

ParcelFormScreen.js:
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { createParcel } from '../actions/parcelActions';
import { View, TextInput, Button } from 'react-native';

const ParcelFormScreen = () => {
  const [formData, setFormData] = useState({
    receiver_phone: '',
    receiver_name: '',
    delivery_type: '',
    description: '',
    weight: '',
    sender_phone: '',
    sender_name: '',
    sender_address: '',
    receiver_address: '',
    sender_city: '',
    receiver_city: '',
  });

  const dispatch = useDispatch();

  const handleChange = (name, value) => {
    setFormData({
      ...formData,
      [name]: value
    });
  };

  const handleSubmit = () => {
    dispatch(createParcel(formData));
  };

  return (
    <View>
      <TextInput   placeholder="Receiver   Phone"   value={formData.receiver_phone}
onChangeText={text => handleChange('receiver_phone', text)} />
      <TextInput    placeholder="Receiver    Name"    value={formData.receiver_name}
onChangeText={text => handleChange('receiver_name', text)} />
      <TextInput    placeholder="Delivery    Type"    value={formData.delivery_type}
onChangeText={text => handleChange('delivery_type', text)} />
      <TextInput       placeholder="Description"       value={formData.description}
onChangeText={text => handleChange('description', text)} />
      <TextInput placeholder="Weight" value={formData.weight} onChangeText={text =>
handleChange('weight', text)} />
      <TextInput    placeholder="Sender    Phone"    value={formData.sender_phone}
onChangeText={text => handleChange('sender_phone', text)} />
      <TextInput     placeholder="Sender     Name"     value={formData.sender_name}
onChangeText={text => handleChange('sender_name', text)} />
      <TextInput   placeholder="Sender   Address"   value={formData.sender_address}
onChangeText={text => handleChange('sender_address', text)} />
      <TextInput  placeholder="Receiver  Address"  value={formData.receiver_address}
onChangeText={text => handleChange('receiver_address', text)} />
      <TextInput     placeholder="Sender     City"     value={formData.sender_city}
onChangeText={text => handleChange('sender_city', text)} />
```

```
      <TextInput   placeholder="Receiver   City"   value={formData.receiver_city}
onChangeText={text => handleChange('receiver_city', text)} />
      <Button title="Submit" onPress={handleSubmit} />
    </View>
  );
};

export default ParcelFormScreen;
```

В.3. Використання Axios для взаємодії з API (веб-додаток)

ParcelActions.js:

```
import axios from 'axios';

export const fetchParcels = () => async dispatch => {
  const response = await axios.get('/api/parcels');
  dispatch({ type: 'FETCH_PARCELS', payload: response.data });
};

export const createParcel = (formData) => async dispatch => {
  const response = await axios.post('/api/parcels', formData);
  dispatch({ type: 'CREATE_PARCEL', payload: response.data });
};
```