МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

Виконав: студент групи 4151

Дрозд В. С.

(підпис, ПІБ)

Керівник роботи:

доцент, к.ф-м.н., доцент

(посада, науковий ступень, вчене звання)

Латанська Л. О.

(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

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

Студенту Дрозду Владиславу Сергійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

Керівник роботи Латанська Людмила Олексіївна, к.ф-м.н., доцент

Затверджені наказом ректора №256-уч від «11» травня 2022 р.

2. Термін подання роботи: 10.06.2022 р.

3. Вихідні дані по роботі:

4. Перелік питань, що належать до розробки (найменування розділів) Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки(технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів

1. Титульний слайд, 2. Мета роботи, 3. Організаційна структура логістичного підприємства «МістЕкспрес», 4. Програми-аналоги та обґрунтування розробки, 5. Постановка задачі на розробку, 6. Засоби розробки, 7. Модель варіантів використання, 8. Діаграма діяльності, 9. Концептуальна модель даних, 10. Фізична модель даних, 11. Архітектура, 12. Статична модель, 13. Динамічна модель, 14. Діаграма компонентів, 15. Діаграма розгортання, 16-19. Користувацький інтерфейс, 20. Висновки.

6. Консультант розділів роботи

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

7. Дата видачі завдання 11.05.2022 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Термін виконання	Примітка
1.	Підготовка розділу Вступ	24.03.2022	ПП*
2.	Опис та аналіз предметної галузі	28.03.2022	ПП*
3.	Постановка задачі	31.03.2022	ПП*
4.	Ескізний проєкт програмного забезпечення	27.04.2022	
5.	Технічний проєкт програмного забезпечення	06.05.2022	
6.	Робочий проєкт програмного забезпечення	13.05.2022	
7.	Підготовка розділу Результати розробки програмного забезпечення	17.05.2022	
8.	Підготовка розділу з охорони праці	18.05.2022	ПП*
9.	Підготовка розділу Висновки	19.05.2022	
10.	Оформлення списку використаних джерел та додатків	20.05.2022	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2«Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від20.01.2020 р.)	10.06.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	12.06.2022	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	14.06.2022	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	24.06.2022	

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

Студент _____ Дрозд В. С.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л. О.
(підпис) (ПІБ)

АНОТАЦІЯ

В кваліфікаційній роботі розв’язується практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

При реалізації програмного забезпечення буде використовуватися остання стабільна версія програмного забезпечення Intellij Idea. Для розробки серверної частини буде використовуватися технологія Java Enterprise Edition, для розробки клієнтської частини веб-фреймворк AngularJS. Для зберігання даних буде використана реляційна база даних PostgreSQL.

Робота викладена на 124 сторінках друкованого тексту, містить 29 рисунків, 11 таблиць, 5 додатків та список використаних джерел з 13 найменувань.

ABSTRACT

The qualification work solves the practical problem of software engineering, characterized by complexity and uncertainty of conditions, namely the development of software to automate the dispatch department of the transport company "MistExpress".

The latest stable version of Intellij Idea software will be used when implementing the software. Java Enterprise Edition technology will be used to develop the server part, and the AngularJS web framework will be used to develop the client part. A PostgreSQL relational database will be used to store the data.

The work is presented on 124 pages of printed text, contains 29 figures, 11 tables, 5 appendices and a list of used sources from 13 titles.

ЗМІСТ

ВСТУП

Автомобільний транспорт в Україні представляє найбільш поширений та гнучкий вид транспорту. Сфера застосування автомобільного транспорту широка і включає в себе міжнародні, міжрегіональні та регіональні перевезення, короткі міські перевезення, доставку вантажів до залізничних станцій і річкових пристаней та до кінцевого споживача. Щодня автомобільним транспортом в Україні перевозиться у середньому 3303 тис. тон, що складає 73 відсотка від усіх перевезень усіма іншими видами транспорту [1]. Якщо порівнювати з аналогічним показником залізничного транспорту, то це в 6 разів більше по обсягу вантажних перевезень.

Вантажоперевезення найбільш “ринковий” сектор економіки, який залежить від постійно змінних вантажопотоків, які генеруються ринками товарів і послуг, що постійно розвиваються, тому, щоб завжди залишатися прибутковими, автотранспортні підприємства мають використовувати найсучасніше програмне забезпечення управління бізнесом, тому що воно значно полегшує контроль за виконанням різноманітних задач, ведення обліку та інвентаризації. На автотранспортному підприємстві може бути одна, або кілька інформаційних систем, які слугують для оптимізації різноманітних бізнес-процесів, починаючи з тих, які притаманні більшості підприємствам: бухгалтерія, відділ кадрів, адміністрування та управління ресурсами, управління відносинами з клієнтами (CRM), планування ресурсів підприємства (ERP); та тими, які необхідні для продуктивної роботи транспортних підприємств: облік транспортних активів, диспетчерська, система управління транспортом (TMS), технічне обслуговування. Дане програмне забезпечення є інструментом, що дозволяє вести облік та зберігати інформацію про учасників тих чи інших процесів, так і інформацію про процеси в цілому. Перевагами таких сучасних логістичних систем є наявність аналітичних модулів, це у свою чергу дає користувачам можливість

аналізувати та вибрати найбільш вигідні підходи ведення бізнесу та найперспективніших партнерів для ведення подальшої співпраці. Також є рішення, які мають більш складну функціональність, що дозволяє в режимі реального часу, використовуючи систему GPS відстежувати та коригувати маршрути перевезень вантажів. Новітні системи здатні побудувати логістичний ланцюжок в автоматичному режимі не залучаючи при цьому користувача. Проте варто враховувати, чим вищою є функціональність системи, тим вищою буде ціна ліцензії для користувача.

Ринок сучасних програмних рішень для автотранспортних компаній пропонує різноманітне програмне забезпечення, яке можна охарактеризувати за кількома категоріями: готові ERP платформи для бізнесу, які охоплюють усі процеси на підприємстві та які налаштовуються та перепрограмуються для потреби конкретної предметної галузі; SaaS програмне забезпечення; комплекс програм Microsoft Office; розробка десктопного або веб програмного забезпечення з нуля для окремого підприємства; або найбільш розповсюджене – комбіновані програмні рішення, тобто окремі відділи підприємства окремо використовують найбільш ефективні програми.

Важливо розуміти, що маловірогідно створити ефективне програмне забезпечення, яке б ідеально поєднувало все і одразу. Так, якщо перед транспортним підприємством стоїть задача отримати актуальні залишки автозапчастин з майстерні, то підприємству знадобиться програмне забезпечення для складського обліку, бухгалтерії знадобляться програми для автоматизації руху коштів та звітності, диспетчерам та водіям знадобиться інше програмне забезпечення для управління логістичними процесами та автоматизації роботи з замовленнями на перевезення і так далі.

Виходячи з вищесказаного для підприємства краще обмежитися програмами, які призначені для вирішення поставленої задачі, а не переслідувати універсальність та широкий набір функцій, тому що спроби об'єднати в собі занадто багато напрямків, знижують якість та гнучкість роботи з кожним окремо взятим завданням.

Тому метою даної кваліфікаційної роботи є розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства "МістЕкспрес".

Розробка програмного забезпечення в межах даної кваліфікаційної роботи передбачає детальний аналіз предметної галузі з метою виявлення вимог до програмного забезпечення. Буде проведений аналіз існуючих програм аналогів для обґрунтування доцільності і постановки задачі на розробку.

Після етапу збору вимог до програмного забезпечення буде виконаний ескізний проект програмного забезпечення, який буде описувати варіанти використання та їх специфікації. Буде розроблена концептуальна модель даних та проект користувацького інтерфейсу.

За результатами завершення ескізного проекту буде виконаний технічний проект програмного забезпечення, що включає в себе архітектуру програмного забезпечення, статичну модель програмного забезпечення та її специфікацію, динамічну модель та логічну модель даних.

На основі попереднього етапу буде розроблений робочий проект програмного забезпечення, в якому буде обґрунтування вибору технології програмування, фізична модель даних, діаграма розгортання. За результатами вибору засобів реалізації буде проведено кодування, тестування та випробування програмного забезпечення

Результатом кваліфікаційної роботи має стати веб-застосунок, призначений для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

1 АНАЛІЗ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС» ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЇЇ АВТОМАТИЗАЦІЇ

1.1 Аналіз організаційної структури логістичного підприємства та роботи диспетчерського відділу

Підприємство «МістЕкспрес» займається вантажоперевезеннями вантажним автомобільним транспортом по Україні. Підприємство складається з відділів управління, фінансів, автопарку та диспетчерської. Відділ управління займається організацією і розробками стратегій розвитку, та в свою чергу підрозділяється на саму адміністрацію, відділ кадрів та відділ технічної підтримки. Відділ фінансів займається контролем правильності ведення підприємницької діяльності підприємства згідно законів. Аудит займається перевіркою роботи усього бухгалтерського відділу. Відділ автопарку оперує основним активом логістичного підприємства – вантажівками та трейлерами, також займається їх ремонтом та обслуговуванням. Диспетчерський відділ спілкується з клієнтами та укладає з ними договори на перевезення вантажів тим самим даючи завдання на перевезення водіям.

Перевезення вантажів є основною послугою логістичного підприємства і саме диспетчери керують цим процесом. Вони консультують та домовляються з клієнтами, планують роботу водіїв та транспорту, стежать за виконанням замовлень постійно знаходячись в контакті з підпорядкованими водіями.

На рисунку 1.1 представлена організаційна структура логістичного підприємства «МістЕкспрес».

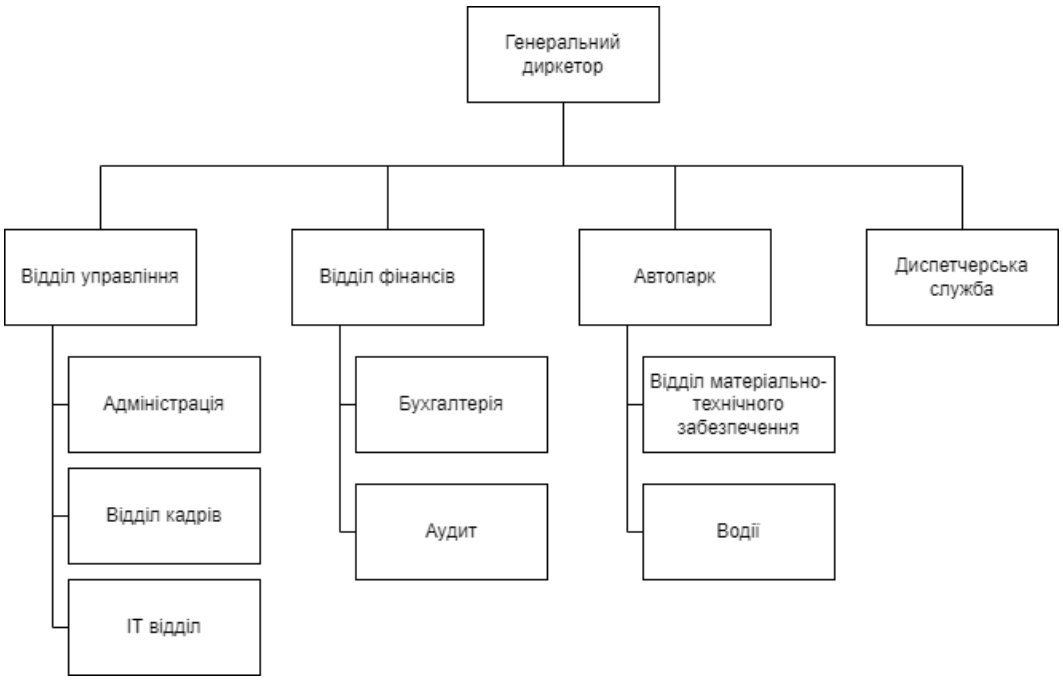


Рисунок 1.1 – Організаційна структура підприємства

Далі буде описаний бізнес-процес опрацювання диспетчером замовлення на перевезення. Основними суб`єктами транспортної угоди є Клієнт – замовник транспортної послуги та Перевізник – її виконавець. Клієнтом може виступати Вантажовідправник, Вантажоотримувач, або інша зацікавлена в перевезенні особа. Основними дійовими особами Перевізника є Водій, який бере участь у перевезенні вантажу вантажівкою компанії, та Диспетчер – центральна фігура, що координує дії усіх інших учасників транспортної угоди, а також веде супровідну документацію.

Комунікація та обмін документами між усіма учасниками транспортної угоди відбувається через мобільний зв`язок, або електронну пошту, так як усі вони можуть знаходитися на великій відстані один від одного.

Для того, щоб знайти виконавців або клієнтів, клієнт публікує оголошення про наявні вантажі та Перевізник публікує оголошення про вільний транспорт на спеціальних онлайн транспортних біржах (найпопулярніша в Україні https://lardi-trans.com), для того, щоб Перевізник зміг знайти вантаж, щоб завантажити свої вантажівки та отримувати прибуток, а Клієнт перевізника який найшвидше надасть йому транспортну послугу.

Тобто, немає значення хто перший буде звертатися за послугою. Далі бізнес алгоритм створення та виконання транспортної угоди буде виглядати наступним чином:

1) Диспетчер та Клієнт узгоджують усі нюанси перевезення вантажу.

2) Диспетчер та Клієнт підписують документ «Договір про надання транспортних послуг», в якому описані всі деталі перевезення і права та обов`язки сторін.

3) Диспетчер інформує Водія про нове транспортне замовлення та надсилає йому усю необхідну інформацію про Клієнта, вантаж, маршрут.

4) Водій прибуває на місце завантаження.

5) Вантажовідправник створює документ «Товарно-транспортна накладна» (далі «ТТН») і передає його Водієві. Цей документ підтверджує законність перевезення даного вантажу для контролюючих органів.

6) Водій доставляє вантаж до пункту розвантаження.

7) Вантажоотримувач отримує свій екземпляр ТТН.

8) Диспетчер складає рахунок перевізника не пізніше трьох днів після виконання перевезення із додаванням ТТН. Цей документ описує реквізити та порядок оплати для Клієнта.

9) Клієнт оплачує послуги перевезення вантажу.

10) Замовлення можна вважати виконаним.

11) Диспетчер складає «Маршрутний лист» для підтвердження списання ПММ (паливно-мастильних матеріалів). Згодом цей документ опрацьовується бухгалтерами для підтвердження списання коштів для податкової.

1.2 Аналіз вимог до програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства

Виконавши аналіз організаційної структури логістичного підприємства та роботи диспетчерського відділу для обґрунтування доцільності розробки був виділений набір основних високорівневих бізнес-вимог до розробляємого

програмного забезпечення. За словами Карла Вігерса Бізнес вимоги містять високорівневі цілі організації чи замовників системи. У цьому документі пояснюється, чому організації потрібна така система, тобто описані цілі, які організація має намір досягти за її допомогою. Також ці вимоги являються границями проекту [2].

Опис користувачів.

Адміністратор – працівник підприємства, користувач програмного забезпечення, відповідальний за адміністрування програмного забезпечення, тобто за інформацію про активи підприємства, та за управління правами доступу інших користувачів.

Диспетчер – працівник підприємства, користувач програмного забезпечення, в обов’язки якого входить робота з замовленнями на перевезення від клієнтів, призначення водіїв і транспорту для виконання заявок, супровід документації по кожній заявці.

Водій – працівник підприємства, користувач програмного забезпечення, який керує транспортним засобом задля виконання заявок на перевезення. Використовує програмне забезпечення для актуалізації інформації про перевезення.

Перелік вимог.

Програмне забезпечення повинно:

– дозволяти диспетчеру опрацьовувати замовлення на перевезення, тобто зберігати та редагувати їх, призначати замовленням різні статуси під час їх виконання;

– надати можливість контролювати рух вантажівок в режимі реального часу за допомогою інтеграції з GPS провайдером трекінгової системи;

– вести облік клієнтів та їх замовлень;

– дозволяти диспетчеру назначити незадіяних водія та транспорт для виконання замовлення на перевезення;

– дозволяти диспетчеру створювати в системі клієнтів для напрацювання клієнтської бази;

– дозволяти диспетчеру зберігати місця, пов’язані з клієнтом;

– дозволяти на основі заповнених диспетчером замовлення та рейсу генерувати необхідні для дотримання законів в області вантажних перевезень супровідні документи: “Договір про надання транспортних послуг”, “ТТН”, “Маршрутний лист”;

– контролювати виплати та заборгованості від клієнтів за надані послуги;

– інформувати про зайнятість водіїв та транспорту для того, щоб диспетчер завжди міг планувати рейси на майбутнє;

– дозволити зробити вибір типу перевезення: збірне (у складі збірного вантажу), або доставка окремим транспортом;

– надати можливість диспетчеру створювати рейси з кількох замовлень на перевезення;

– передавати інформацію про поточний рейс на сторінку водія;

– надати можливість водію міняти статуси рейсу та завершити його;

– по закінченню рейсу генерувати документ “Маршрутний лист” для списання бухгалтерією паливно-мастильних матеріалів;

– надати адміністратору можливість отримувати звіти з роботи диспетчерського відділу;

– надати адміністратору можливість реєструвати диспетчерів та водіїв в програмі і надавати їм права доступу.

1.3 Аналіз існуючих аналогів та обґрунтування доцільності розробки програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства

На даний момент існує невелика кількість конкуруючого програмного забезпечення для автотранспортних підприємств. Зупинимося на кількох

найбільш популярних програмах, що використовуються на українському ринку вантажоперевезень, таких як «USUSoft», «ITOB», «РемОнлайн».

1.3.1 Веб-застосунок «РемОнлайн»

РемОнлайн використовує інший підхід для доставки свого програмного забезпечення до кінцевого користувача, ніж обидві наступні компанії з розробки програмного забезпечення. USUSoft та ITOB продають ліцензію на використання програмного забезпечення та допомагають з впровадженням системи на підприємство. РемОнлайн надає своє програмне забезпечення по підписці за SaaS моделлю. РемОнлайн має меншу функціональність, що не включає основних можливостей, які є дуже важливими для логістичного, наприклад генерація супровідної документації та трекінг транспорту в режимі реального часу, для цих цілей автотранспортне підприємство повинно буде шукати сторонні рішення. Можливість інтеграції теж обмежена, тому що усі клієнти підписники використовують одне програмне забезпечення на сервері компанії РемОнлайн, тому зміна програмного коду буде впливати на всіх клієнтів. Тож програмне забезпечення не може бути налаштоване для конкретного підприємства.

Також дана схема надання програмного забезпечення дуже вразлива до несанкціонованого доступу, тому що дані усіх клієнтів містяться в одній базі даних. Але для малих автотранспортних підприємств до 10 диспетчерів, ці недоліки перекриваються дуже низькою, в порівнянні з іншими рішеннями, вартістю обслуговування та відсутністю необхідності розгортання власного серверу, тому що вся відповідальність за технічне обслуговування повністю лежить на компанії провайдера програмного забезпечення.

Приклад інтерфейсу основної сторінки для роботи з замовленнями представлений на рисунку 1.2.

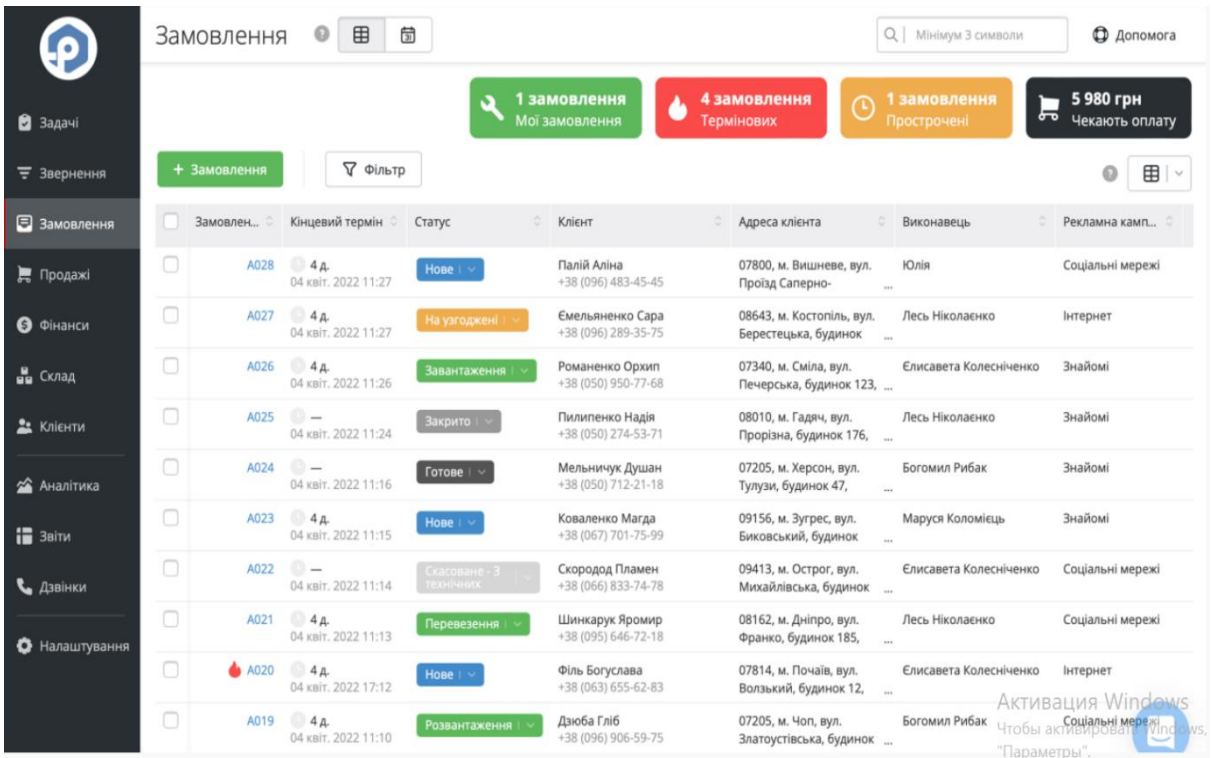


Рисунок 1.2 – Інтерфейс програми РемОнлайн [3]

Можна виділити наступні недоліки розглянутого ПЗ:

– неможливість налаштувати ПЗ під бізнес-процеси підприємства;

– неможливість інтеграції із вже наявним ПЗ на підприємстві;

– відсутність важливого для диспетчерської функціоналу;

– зберігання даних на сервері провайдера ПЗ.

1.3.2 Комплекс програм «ІТОВ»

Програмне забезпечення ІТОВ побудоване на платформі 1С:Підприємство 8. Функціональні можливості системи охоплюють весь контур завдань управління логістикою та ланцюгами постачання. Типовий функціонал можна налаштувати відповідно до потреб користувачів та адаптувати під завдання практично будь-якої компанії – залежно від специфічних особливостей та масштабу діяльності. Це дозволяє максимально використовувати можливості програми, підвищуючи ефективність управління процесами перевезень.

Приклад інтерфейсу вікна програми наведений на рисунку 1.3.

Недоліками є складність з впровадженням та супроводом програмного забезпечення, а також проблемність налаштування віддаленого доступу для роботи онлайн. Перевагами даного програмного забезпечення є найширший з представлених перелік функціональних можливостей та легка інтеграція з іншим програмним забезпеченням платформи 1С. Програма розроблена для ринку країн СНД, тому при її адаптації для Українського підприємства потрібно вносити зміни в код окремих модулів програми.

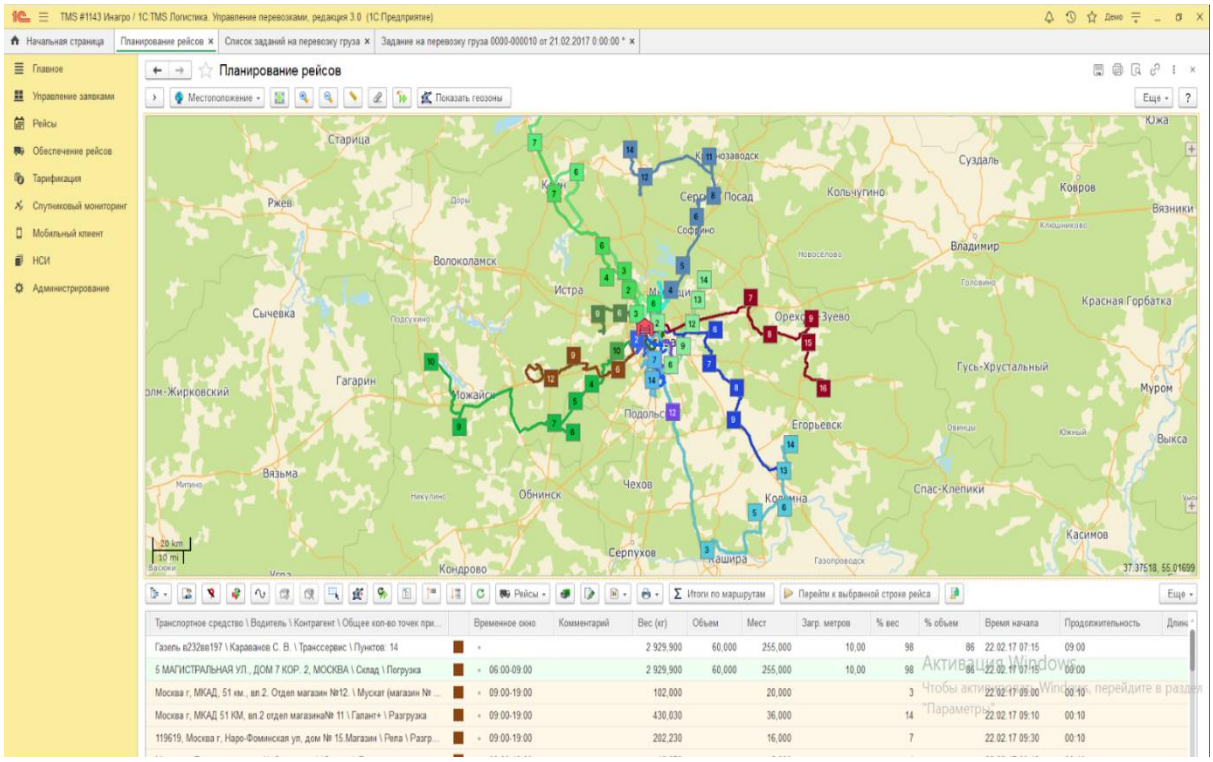


Рисунок 1.3 – Інтерфейс програми ITOB [4]

Можна виділити наступні недоліки розглянутого ПЗ:

– складність в налаштуванні ПЗ для конкретного підприємства;

– велика кількість багів через надмірний функціонал;

– важкість в налаштуванні віддаленої роботи персоналу.

1.3.3 Десктопний застосунок «USUSoft»

Програмне забезпечення для автотранспортних підприємств від компанії розробника «USUSoft» позиціонує себе, як десктопну програму для операційної системи Windows, що створена для автоматизації окремого бізнес-процесу опрацювання заявок на перевезення. Програма має функціонал, що дозволяє диспетчеру працювати з замовленнями, формувати з них маршрути та напрацьовувати клієнтську базу.

Основними недоліками програми є відсутність можливості роботи з документами, складність розгортання і оновлення програмного забезпечення та ускладнена можливість віддаленої роботи, так як, якщо це десктопне рішення, то воно буде встановлене на робочому комп'ютері в офісі, що унеможливлює роботу з дому, що в останній час є дуже актуальним.

Приклад інтерфейсу вікна програми наведений на рисунку 1.4.

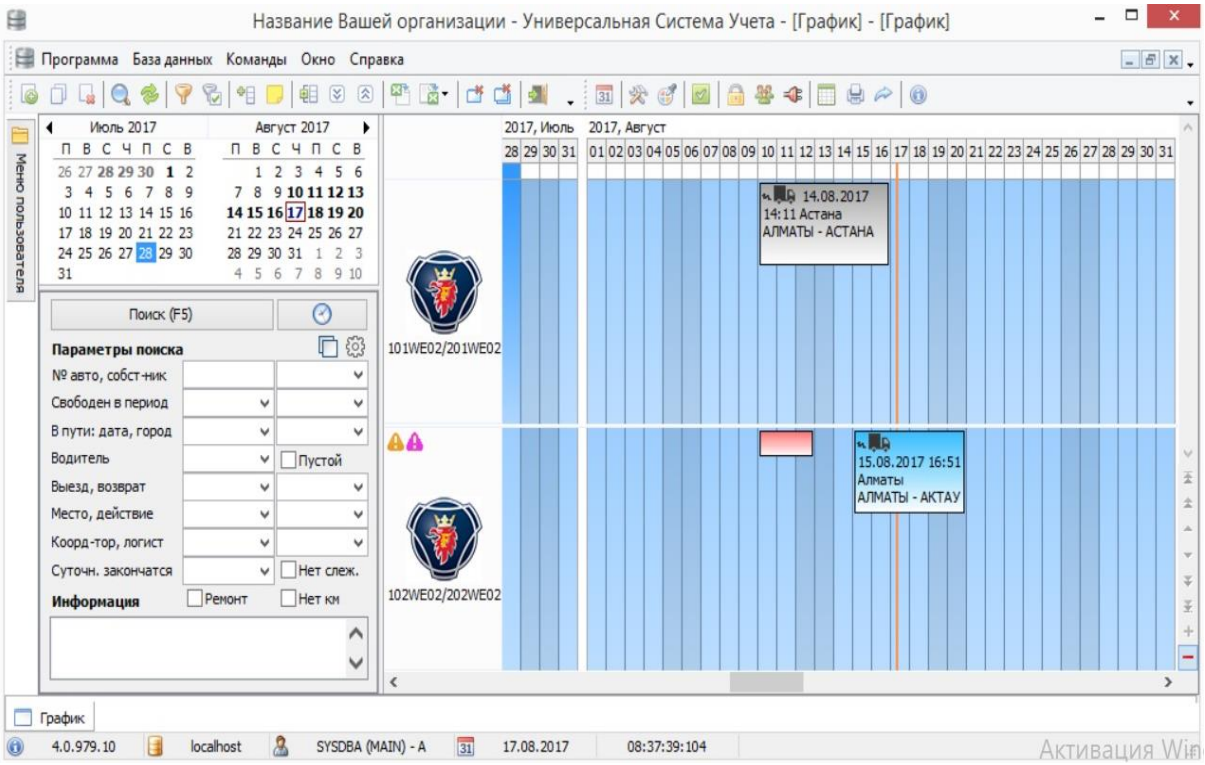


Рисунок 1.4 – Інтерфейс програми USUSoft [5]

Можна виділити наступні недоліки розглянутого ПЗ:

– відсутність важливого для диспетчерської функціоналу;

– складність розгортання програмного забезпечення;

– важкість в налаштуванні віддаленої роботи персоналу.

1.3.4 Обґрунтування доцільності розробки програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства

Розглянувши існуючі варіанти програмного забезпечення, яке реалізує функціонал диспетчерської, стала зрозуміла необхідність створення власної програми, яка б була цілком побудована для вирішення задачі диспетчеризування для логістичної компанії «МістЕкспрес». Програмне забезпечення, що проектується, відрізняється від існуючих аналогів тим, що по-перше, воно буде мати всі необхідні модулі для того, щоб цілком розв'язувати поставлену бізнес задачу без необхідності інтеграції з прикладним програмним забезпеченням, яке б розв’язувало вузьконаправлені підзадачі, по-друге, задумане програмне забезпечення буде легке в інтеграції, супроводженні та розширюваності, тому що воно буде побудоване на основі клієнт-серверної архітектури, по-третє при програмуванні програмного забезпечення, що проектується, будуть використовуватися найсучасніші технології розробки веб-додатків, що позитивно відобразиться на економічній складовій впровадження, тому що кожне готове програмне забезпечення потребує певних доробок та розширення функціоналу під вимоги замовника та інтеграції зі вже працюючим програмним забезпеченням.

1.4 Постановка задачі на розробку програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства

Проаналізувавши предметну галузь вантажних перевезень та дослідивши аналогічне програмне забезпечення для диспетчеризації в автотранспортних підприємствах, проаналізувавши їх сильні та слабкі сторони стало можливим оцінити, які саме проблеми необхідно подолати в нашому програмному забезпеченні.

Необхідно створити веб-застосунок, який би повністю вирішував бізнес задачі диспетчерського відділу.

Згідно з аналізом вимог до програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства встановимо перелік вимог, яким повинен відповідати веб-застосунок, що розробляється.

1) Вимоги до функціональних характеристик:

– контроль руху вантажівок в режимі реального часу за допомогою інтеграції з GPS провайдерами трекінгових систем;

– ведення обліку клієнтів та їх замовлень;

– призначення незадіяних водіїв та транспорт для виконання замовлення на перевезення;

– створення в системі клієнтів для напрацювання клієнтської бази;

– планування рейсів на майбутнє;

– можливість на основі заповнених диспетчером замовлення та рейсу автоматично генерувати необхідні для дотримання законів в області вантажних перевезень супровідні документи: “Договір про надання транспортних послуг”, “ТТН”, “Маршрутний лист”;

– контроль за виплатами та заборгованостями від клієнтів за надані послуги;

– надання можливості працювати зі збірними вантажами;

– створення рейсів з кількох заявок на перевезення;

– передача інформації про рейси на сторінку водія;

– надання водію можливості змінювати інформацію про рейс під час його виконання;

– автоматична генерація документа “Маршрутний лист” для списання бухгалтерією паливно-мастильних матеріалів;

– отримання звітів з роботи диспетчерського відділу;

– реєстрація диспетчерів та водіїв в програмі і керування їхніми правами доступу;

Реалізація комплексу перелічених функціональних вимог має вирішувати усі задачі диспетчерського відділу логістичного підприємства.

2) Вимоги до організації вхідних та вихідних даних

Вхідними даними для роботи диспетчера, адміністратора та водія з програмним забезпеченням є текстова інформація, що вводиться в поля вводу та форми на веб сторінках браузерного клієнта веб-застосунку та зберігаються до бази даних.

Вихідні дані відображаються користувачу на веб-сторінках браузера, також вихідними даними є супровідна документація та звіти, що зберігаються на диску.

3) Вимоги до складу та параметрів технічних засобів:

Для роботи програмного забезпечення необхідна наявність у користувачів смартфону та персонального комп’ютеру, який має такі мінімальні характеристики:

– 8 Гб оперативної пам’яті;

– 150 Гб вільного дискового простору;

– інтегрована в процесор відеокарта;

– дисплей розміром 1536x864 пікселів;

– клавіатура та миша;

Для розгортання програмного забезпечення на сервері, який здатен опрацьовувати 50 активних користувачів необхідні два комп’ютера . Сервер веб-застосунку повинен мати такі мінімальні характеристики:

– 16-ти поточний процесор;

– тактова частота процесора 3 ГГц;

– 16 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

Сервер бази даних повинен мати такі мінімальні характеристики:

– 4-х поточний процесор;

– тактова частота процесора 2.5 ГГц;

– 8 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

4) Вимоги до інформаційної та програмної сумісності

Для розробки веб-застосунку використати останню стабільну версію програмного забезпечення Intellij Idea. Для розробки серверної частини використати технологію Java Enterprice Edition, для розробки клієнтської частини веб-фреймворк AngularJS. Для зберігання даних використати реляційну базу даних PostgreSQL.

2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

2.1 Ескізний проект програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

Ескізним проектом називають пакет конструкторської документації, який створюється на стадії розробки програмного забезпечення. Мета створення цих документів – встановити принципові, конструктивні рішення, представити їх для ознайомлення з призначенням, принципами роботи, та будови програмного забезпечення, що розробляється [6].

2.1.1 Побудова моделі варіантів використання програмного забезпечення для автоматизації роботи диспетчерського відділу

Діаграма прецедентів в UML, діаграма, на якій зображено відношення між акторами і прецедентами в системі. Діаграма прецедентів є графом, що складається з множини акторів, варіантів використання, обмежених границею системи, асоціацій між акторами і прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами.

Прецедент – це цілісна одиниця видимої ззовні функціональності, яка надається системою. Метою варіанту використання є визначення частини поведінки без розкриття внутрішньої структури системи. Визначення варіанту використання включає всю поведінку, яку він тягне – основні послідовності, різні варіації нормальної поведінки та всі виняткові умови, які можуть виникнути з такою поведінкою, разом із бажаною реакцією [6].

Опис акторів

Визначимо акторів, які будуть взаємодіяти з програмним забезпеченням. Профілі акторів описані в таблицях 2.1, 2.2, 2.3, 2.4.

Актори це користувачі, які взаємодіють із системою. Актором може бути особа, організація або стороння система, яка взаємодіє з додатком або системою. Вони повинні бути зовнішніми об'єктами, які виробляють або споживають дані.

Таблиця 2.1 – Профіль користувача - Адміністратор

Характеристика	Опис
Опис актора	Працівник компанії, що відповідальний за адміністрування системи, тобто за управлінням правами доступу інших користувачів, управлінням інформацією про активи компанії. Наділений правом створювати/редагувати дані про активи компанії; надавати права доступу до системи іншим працівникам.
Тип	Користувач
Відповідальність	Наявність в системі актуальних даних про активи підприємства
Критерій успіху	Вчасно редагувати дані про активи компанії для того, щоб інші працівники працювали з актуальними даними.

Таблиця 2.2 – Профіль користувача Диспетчер

Характеристика	Опис
Опис актора	Працівник компанії, в обов’язки якого входить робота з замовленнями на перевезення від клієнтів, призначення водіїв і транспорту для виконання заявок, супровід документації по кожній заявці. Наділений правом створювати, редагувати замовлення та маршрути. Координує рух водія.
Тип	Користувач
Відповідальність	Ведення замовлень та супровідної документації
Критерій успіху	Правильно планувати і опрацювати замовлення та маршрути

Таблиця 2.3 – Профіль користувача - Водій

Характеристика	Опис
Опис актора	Працівник, який керує транспортним засобом задля виконання заявок на перевезення
Тип	Користувач
Відповідальність	Перевезення вантажу
Критерій успіху	Доставити вантаж у потрібне місце у потрібний час

Таблиця 2.4 – Профіль актора – Компанія провайдер GPS трекінгових послуг

Характеристика	Опис
Опис актора	Компанія, що надає послуги з відслідковування вантажівок через влаштовані GPS маячки
Тип	Зовнішня система
Відповідальність	Технічне обслуговування GPS маячків та збирання та передача телеметрії до бази даних логістичного підприємства «МістЕкспрес»
Критерій успіху	Вчасна та безперебійна передача телеметрії

Опис варіантів використання

Під час дослідження потреб предметної галузі виявлені наступні варіанти використання, які представлені в таблиці 2.5.

Таблиця 2.5 – Виявлені варіанти використання

Код	Актор	Назва	Формулювання
1	2	3	4
АДВ1	Адміністратор Диспетчер Водій	Авторизація	Користувач зможе авторизуватися в системі.
А1	Адміністратор	Отримання звітів	Адміністратор зможе отримувати звіти з роботи компанії.

Продовження таблиці 2.5

1	2	3	4
А2	Адміністратор	Робота з активами підприємства	Додавання, редагування інформації про саму компанію, про її активи, такі як вантажівки, причіпи, термінали.
А3	Адміністратор	Управління правами доступу користувачів до системи	Можливість реєстрації та редагування працівників (водіїв, диспетчерів).
В1	Водій	Робота з призначеним рейсами	Водій матиме список усіх призначених на нього минулих та теперішнього рейсів.
В2	Водій	Редагування рейсу в процесі його виконання	За потреби водій зможе відредагувати та змінити статус рейсу, який він зараз виконує.
Д1	Диспетчер	Контроль руху вантажівок в режимі реального часу	Диспетчер зможе бачити на мапі розташування усіх вантажівок, які на даний момент виконують завдання на перевезення.
Д2	Диспетчер	Робота з замовленнями на перевезення	Формується список усіх замовлень на перевезення. Диспетчер зможе зчитувати зі списку інформацію про актуальні та минулі замовлення.
Д3	Диспетчер	Створення/редагування замовлення на перевезення	Диспетчер має можливість створювати та редагувати замовлення на перевезення.
Д4	Диспетчер	Генерація документів «ТТН» та «Договір на перевезення»	Після збереження замовлення на перевезення можна згенерувати необхідні документи для їх подальшої відправки до замовника послуги.
Д5	Диспетчер	Робота з рейсами	Формується список усіх рейсів. Диспетчер зможе зчитувати зі списку інформацію про актуальні та минулі рейси.
Д6	Диспетчер	Створення/редагування рейсу	Диспетчер має можливість створювати та редагувати рейси.

Продовження таблиці 2.5

1	2	3	4
Д7	Диспетчер	Генерація документу «Маршрутний лист»	Після завершення рейсу диспетчер зможе згенерувати документ «Маршрутний лист» для списання бухгалтерією паливно-мастильних матеріалів.
Д8	Диспетчер	Робота з організаціями контрагентами	Формується список усіх організацій. Диспетчер зможе зчитувати зі списку інформацію про організації.
Д9	Диспетчер	Створення/редагування організації	Диспетчер може занести в систему нову організацію з якою започатковано співробітництво, або відредагувати існуючу.
Д10	Диспетчер	Створення нових місць для організації	Під час роботи з замовленням може знадобитися створити нове місце для завантаження або відвантаження вантажу та прив’язати це місце до існуючої організації.
Д11	Диспетчер	Перегляд дошки зайнятості	Диспетчер може переглянути інтерактивну сторінку на якій вказані проміжки часу коли той чи інший водій та транспорт виконують завдання. Ця дошка потрібна для того, щоб диспетчер міг знайти вільного водія та транспорт і спланувати наступний рейс.
П1	Компанія провайдер GPS трекінгових послуг	Отримання даних від компанії провайдера GPSтрекінгу	Отримання даних телеметрії зі встановлених на вантажівках GPS маячків

Усі вище визначені варіанти використання візуалізовані нижче в діаграмі прецедентів на рисунку 2.1. Дана діаграма показує відносини між акторами та варіанти використання в системі [6]. Варіант використання це опис деякої

послуги, яка надається програмним забезпеченням актору. При цьому діаграма не показує в якому порядку виконуються кроки для досягнення цілі кожного прецедента та яким саме чином реалізована взаємодія між користувачем і системою.

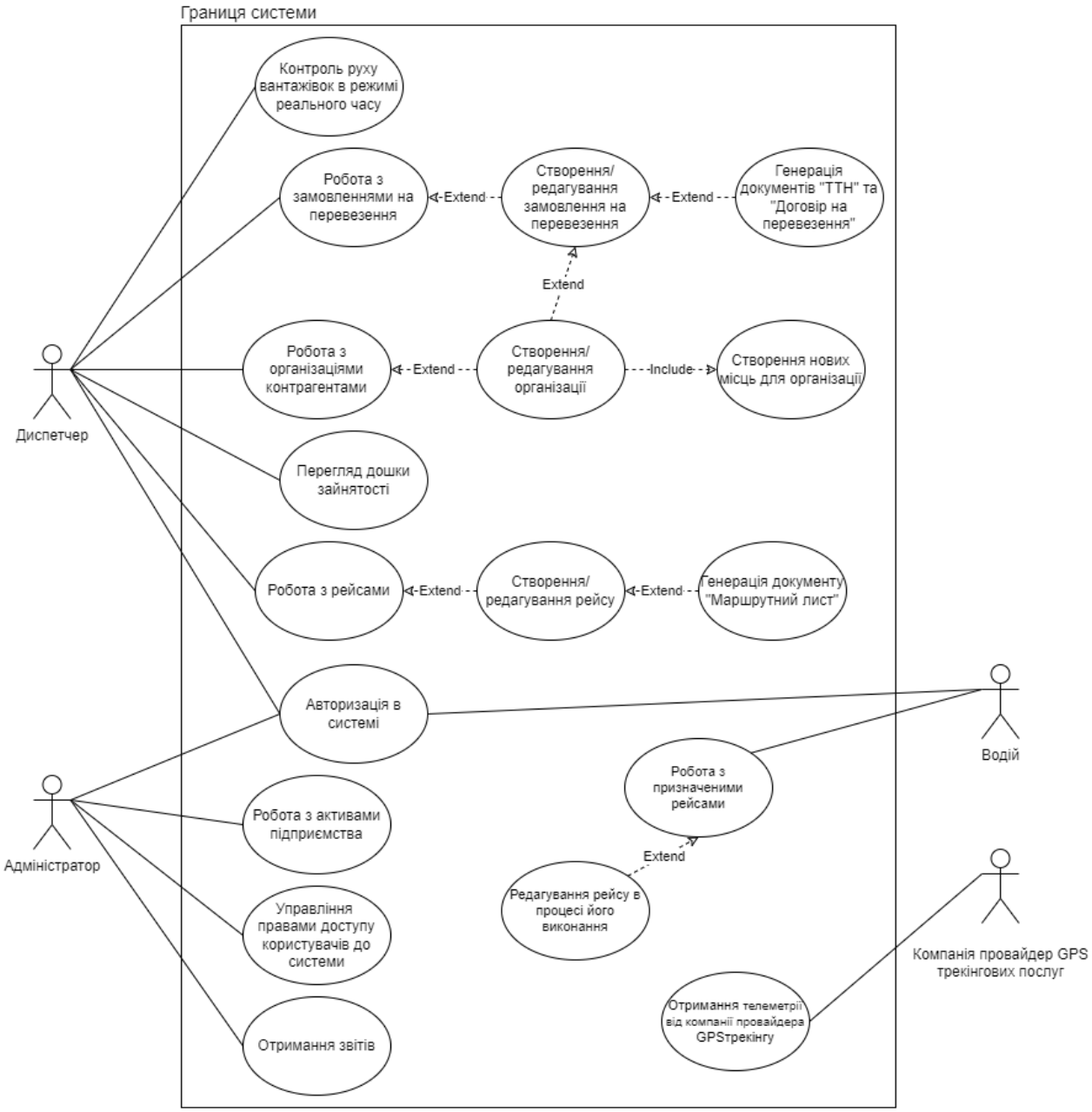


Рисунок 2.1 – Діаграма прецедентів програмного забезпечення для автоматизації роботи диспетчерського відділу

2.1.2 Специфікації основних варіантів використання програмного забезпечення для автоматизації роботи диспетчерського відділу

Специфікації варіантів використання будуть виконані в текстовому та графічному вигляді за допомогою діаграм діяльності. В специфікації будуть використані такі властивості для опису варіанта використання:

– потік подій – опис того, що має виконувати програмне забезпечення щодо даного варіанту використання (не те, яким чином програмне забезпечення вирішує певні питання);

– альтернативний потік подій – опис подій, які можуть відбутися за певних умов, при відхиленні від основного потоку подій;

– попередні умови – опис, що визначає обмеження, що накладаються на систему перед початком виконання варіанта використання;

– вихідні умови – текстовий опис, який визначає обмеження, що накладаються на систему після завершення виконання варіантів використання.

За відсутності властивостей для окремого варіанту використання властивість не буде описана в специфікації.

Специфікація варіанту використання ДЗ «Створення/редагування замовлення на перевезення»

Актор: Диспетчер

Попередні умови:

1) Авторизація в системі.

Потік подій:

1) Перехід на сторінку створення/редагування замовлення.
2) Заповнення даних про клієнта.
3) Заповнення загальних даних про перевезення.
4) Заповнення даних про вантажовідправника і вантажоотримувача.
5) Заповнення даних про вантаж.
6) Додавання вільного водія та транспорту.

7) Автоматична перевірка замовлення.
8) Збереження замовлення на перевезення.

Альтернативний потік подій 1: якщо при заповнені даних про клієнта (2) його не знайдено в базі організацій, то в цій же формі потрібно зберегти нову організацію в систему. Після збереження клієнта перейти до події 3.

Альтернативний потік подій 2: якщо при заповнені даних про вантажовідправника або вантажоотримувача (4) його не знайдено в базі організацій, то в цій же формі потрібно зберегти нову організацію в систему. Після збереження організації перейти до події 5.

Альтернативний потік подій 3: помилка при автоматичній перевірці замовлення (7). В цьому випадку програма видасть повідомлення зі вказівкою на помилку яку потрібно виправити. Після виправлення помилки можна зберегти замовлення.

Альтернативний потік подій 4: якщо потрібно зберегти неповне замовлення, або замовлення з умисною помилкою, то потрібно проігнорувати помилку, яка буде показана після події 7 та зберегти замовлення.

Діаграма діяльності для варіанту використання ДЗ «Створення/редагування замовлення на перевезення» зображена на рисунку 2.2.

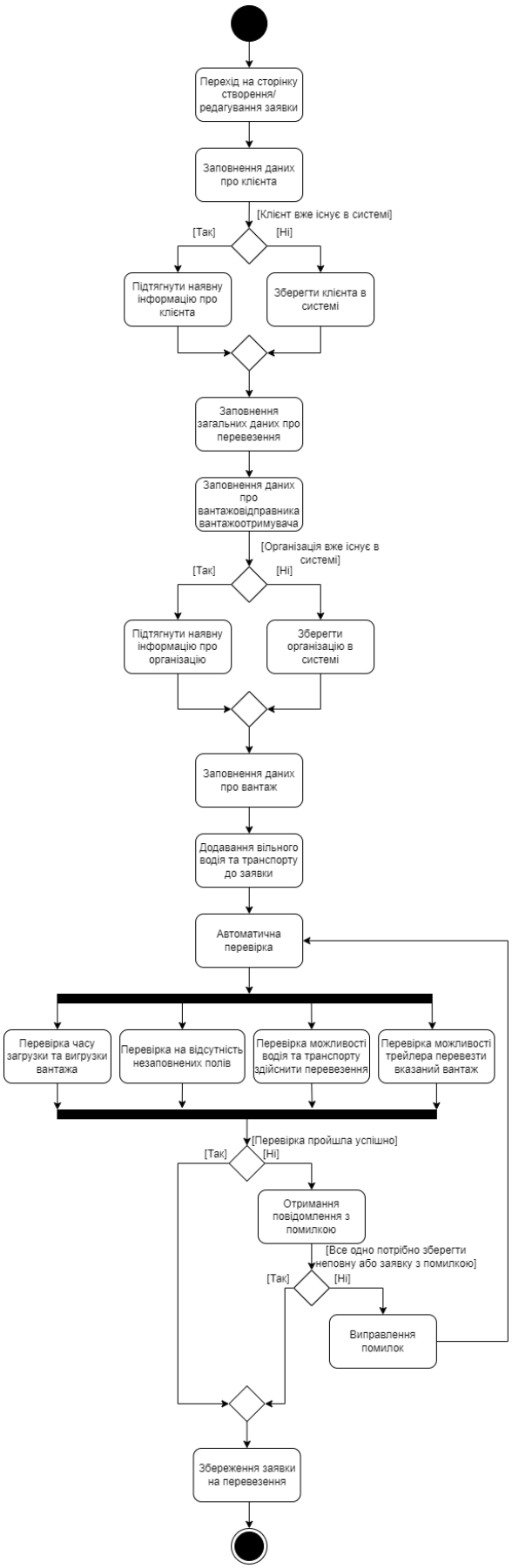


Рисунок 2.2 – Діаграма діяльності для варіанту використання ДЗ «Створення/редагування замовлення на перевезення»

Специфікація варіанту використання Д6 «Створення/редагування рейсу»

Актор: Диспетчер

Попередні умови:

1) Авторизація в системі.
2) Рейс створюється на основі замовлення на перевезення, тому для створення рейсу в базі даних повинно знаходитися хоча б одне не взяте в рейс замовлення на перевезення.

Потік подій:

1) Перехід на сторінку створення/редагування рейсу.
2) Обрання одного замовлення на перевезення для створення нового рейсу.
3) Автоматичне заповнення основної інформації про рейс з обраного замовлення.
4) За потреби додання інших заявок на перевезення до поточного рейсу.
5) Створення точок початку і кінця маршруту.
6) Виставлення точок початку і кінця маршруту, а також усіх точок завантаження і розвантаження з доданих заявок на перевезення в хронологічну послідовність.
7) Автоматична перевірка рейсу.
8) Збереження рейсу.

Альтернативний потік подій 1: якщо рейс буде складатися тільки з одного замовлення на перевезення то подію 4 можна пропустити.

Альтернативний потік подій 2: помилка при автоматичній перевірці рейсу (7). В цьому випадку програма видасть повідомлення зі вказівкою на помилку, яку потрібно виправити.

Діаграма діяльності варіанту використання Д8 зображена на рисунку 2.3.

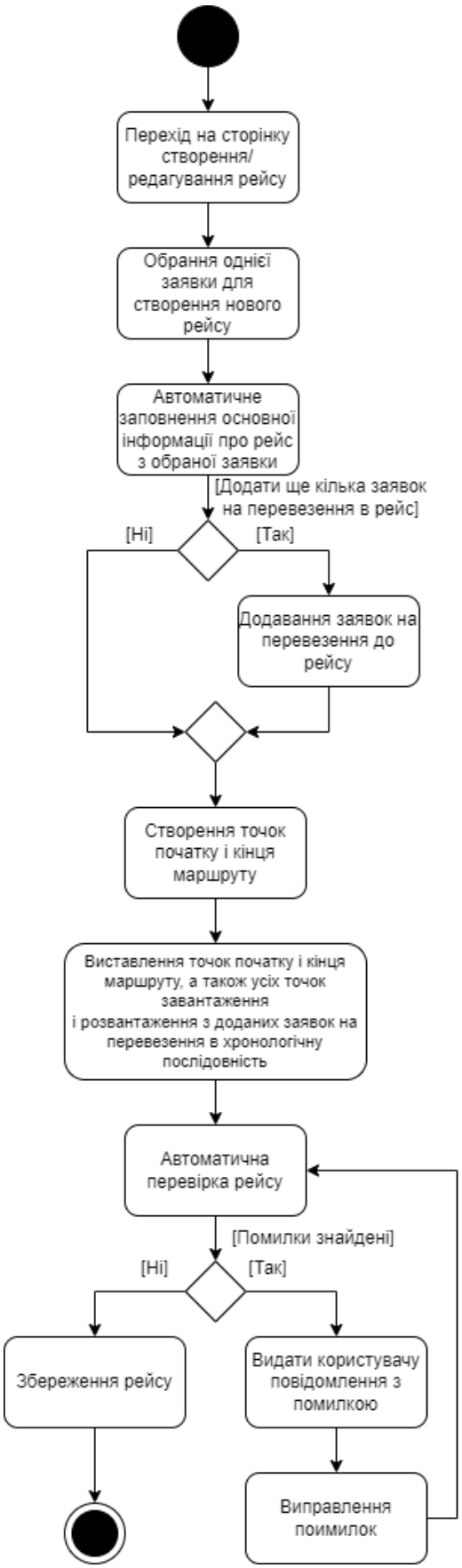


Рисунок 2.3 – Діаграма діяльності для варіанту використання Д6 «Створення/редагування рейсу»

2.1.3 Концептуальна модель даних програмного забезпечення для автоматизації роботи диспетчерського відділу

Це модель даних найвищого рівня, оскільки вона містить найменш деталізовані деталі, але вона встановлює загальний обсяг того, що має бути включено в остаточну фізичну модель даних готового програмного забезпечення. Для цілей описання концептуальної моделі даних слушно використати діаграму «Сутність-Зв’язок». Ця діаграма відображає сутності, їх атрибути та зв’язки між сутностями. Концептуальна модель даних використовується як основа для логічної моделі даних.

Діаграма «Сутність-Зв’язок» наведена на рисунку 2.4, але для початку перелічимо усі сутності та їх атрибути:

1) Користувач.

Атрибути: id, логін, пароль, тип ,статус (активний, обмежений в доступі, видалений), дата реєстрації.

2) Диспетчер.

Атрибути: id, користувач, прізвище, ім’я, по батькові, телефон, email.

3) Водій.

Атрибути: id, прізвище, ім’я, по батькові, користувач, номер водійського посвідчення, відкриті категорії, телефон, email, статус (в рейсі, очікує завдання, вихідний, звільнений).

4) Організація.

Атрибути: id, назва, ПІБ директора, ПІБ контактної особи, телефон, email, місцезнаходження головного офісу.

5) Місце.

Атрибути: id, назва, організація, тип (склад, підприємство, магазин, офіс, гараж), країна, регіон, місто, вулиця, поштовий індекс, довгота, широта.

6) Вантажівка.

Атрибути: id, державний номер, модель, рік, тип пального, одометр, GPSIMEI(номер GPS пристрою), статус (виконує завдання, чекає завдання, на ремонті, виведена з експлуатації).

7) Телеметрія.

Атрибути: час сигналу, IMEI, довгота, широта.

8) Причіп.

Атрибути: id, державний номер, тип (тент, рефрижератор, степдек, трал, насипний), модель, довжина, ширина, висота, максимальне завантаження, статус (виконує завдання, чекає завдання, на ремонті, виведена з експлуатації).

9) Вантаж.

Атрибути: id, замовлення, назва, номер (штрих-код, артикул і т.п.), кількість, розмірність кількості (шт., кг. і т.п.), вид упаковки, температурний режим, вага, довжина, висота, ширина, нотатки.

10)Замовлення.

Атрибути: id, рейс, номер замовлення, диспетчер, водій, вантажівка, причіп, статус (нове, готове до додання в рейс, чекає відправлення в назначений рейс, в рейсі, в дорозі, аварійне, доставлене, виставлений рахунок, оплачене, завершене), дата створення, дата закриття, клієнт, тип (повне, неповне), ціна, ТТН документ, договір документ, нотатки.

11) Зупинка.

Атрибути: id, тип (початок рейсу, завантаження, розвантаження, кінець рейсу), замовлення, рейс, послідовність, організація, місце, час з, час до, контактна персона, контакти, нотатки.

12) Рейс.

Атрибути: id, номер рейсу, диспетчер, водій, вантажівка, причіп, статус (новий, запланований, в процесі виконання, аварійний, закінчений), час початку, час закінчення, початковий одометр, кінцевий одометр, витрачено пального, маршрутний лист документ, нотатки.

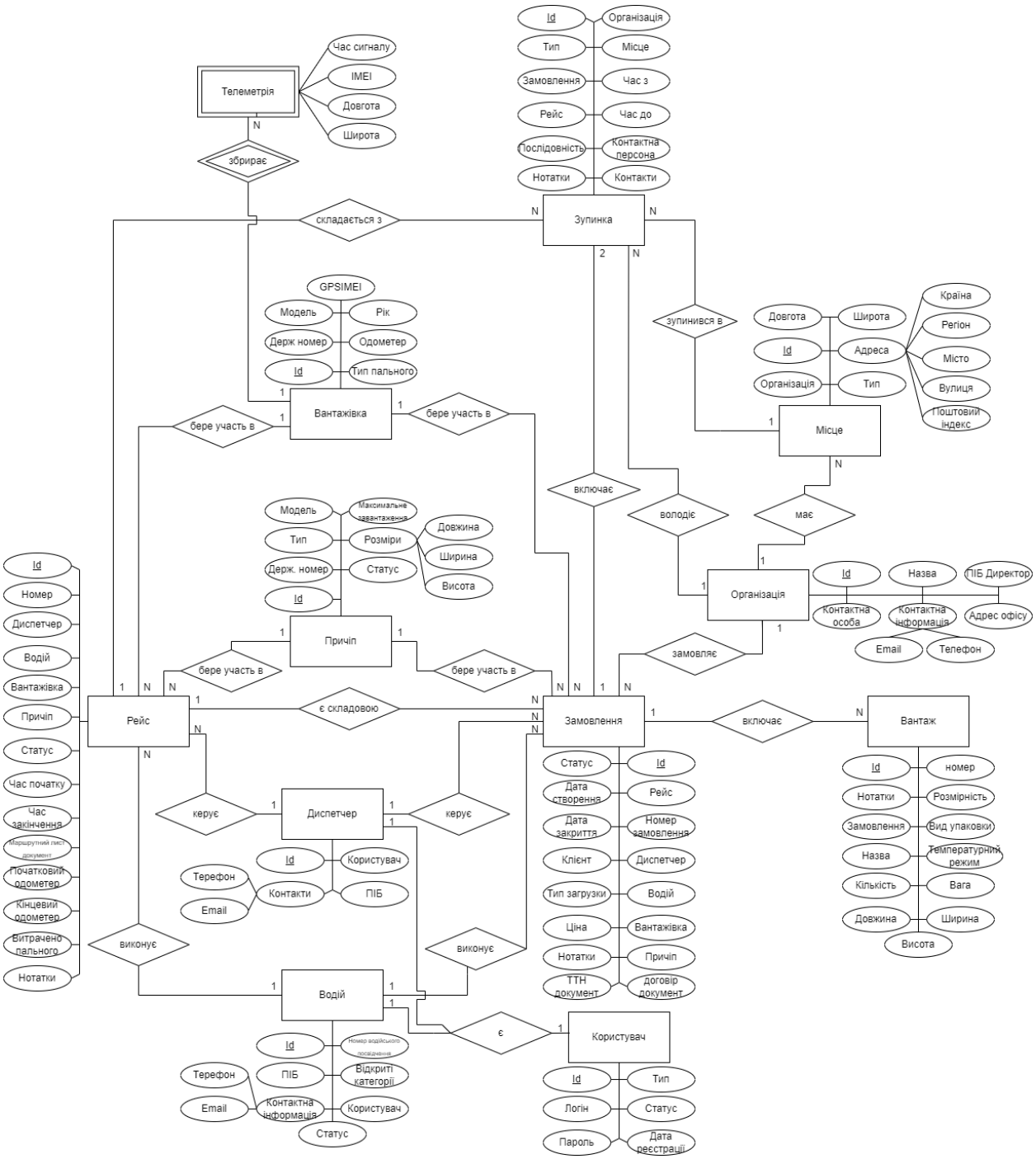


Рисунок 2.4 – Діаграма «Сутність-Зв’язок» програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

2.1.4 Проект користувацького інтерфейсу програмного забезпечення для автоматизації роботи диспетчерського відділу

При розробці користувацького інтерфейсу на першому етапі створюється модель інтерфейсу, що відображає лише розміщення елементів на екрані та їх відносний розмір, а не кінцевий зовнішній вигляд, стилі, колір, тощо.

Наведемо приклад розроблених моделей інтерфейсів для веб-застосунку для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес». Проект сторінки логіну наведений на рисунку 2.5, проект головної сторінки з мапою транспорту наведений на рисунку 2.6, проект сторінки зі списком замовлень наведений на рисунку 2.7, проект сторінки для створення/редагування замовлення на перевезення наведений на рисунку 2.8.

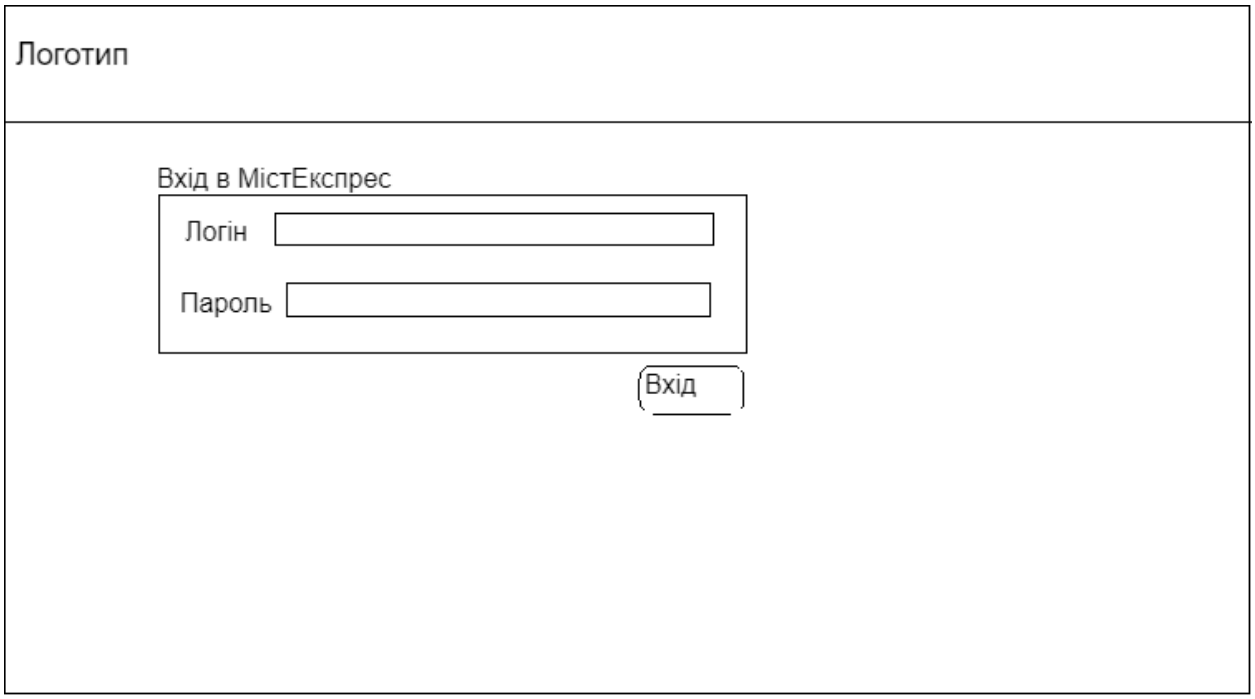


Рисунок 2.5 – Проект сторінки авторизації веб-застосунку

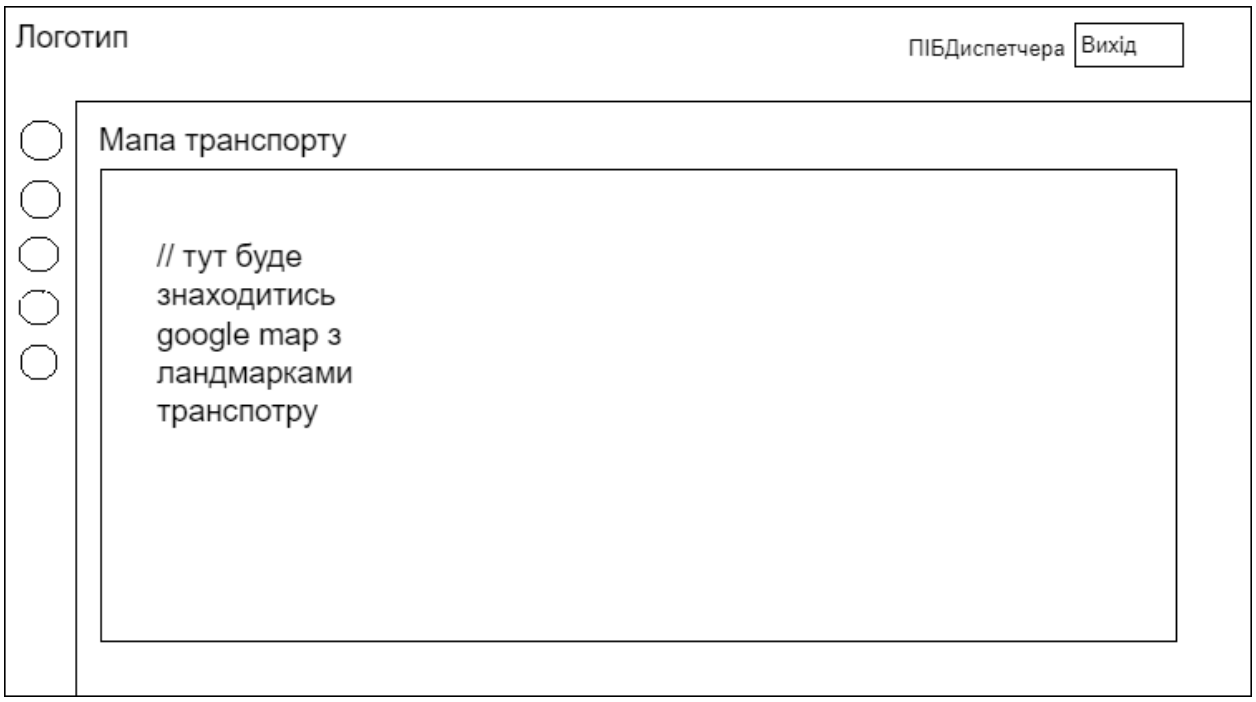


Рисунок 2.6 – Проект сторінки веб-застосунку з мапою транспорту

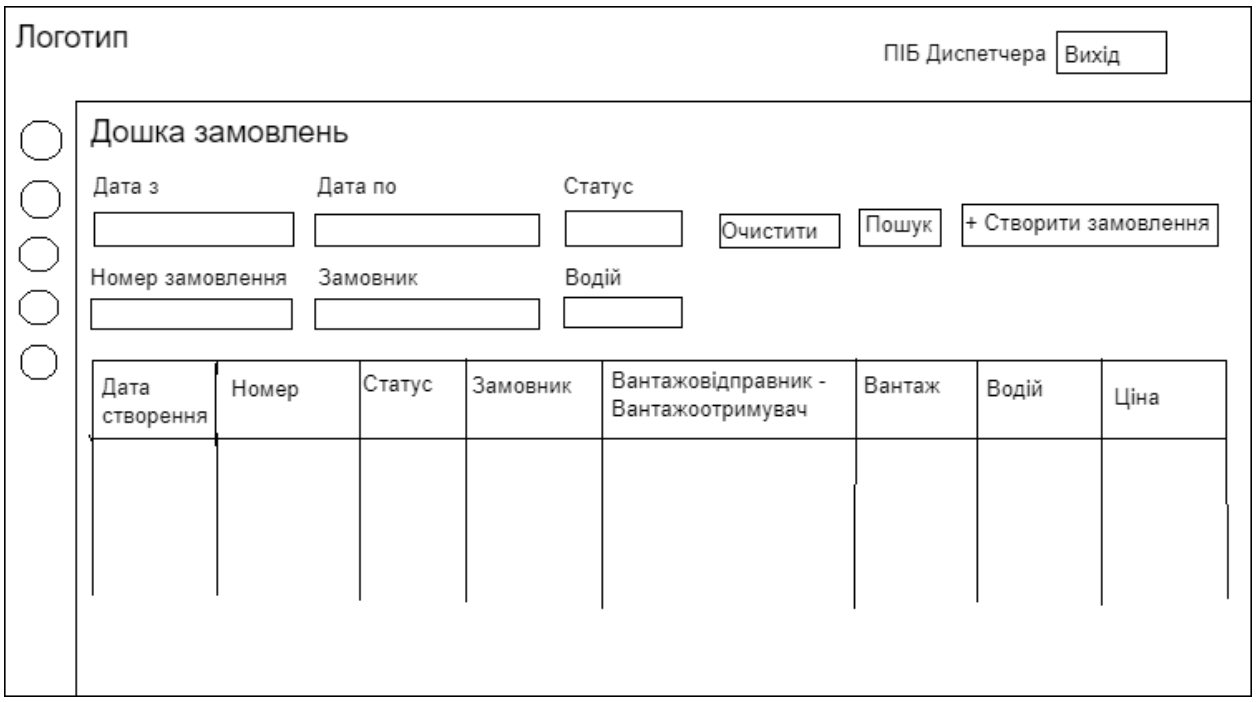


Рисунок 2.7 – Проект сторінки веб-застосунку зі списком замовлень

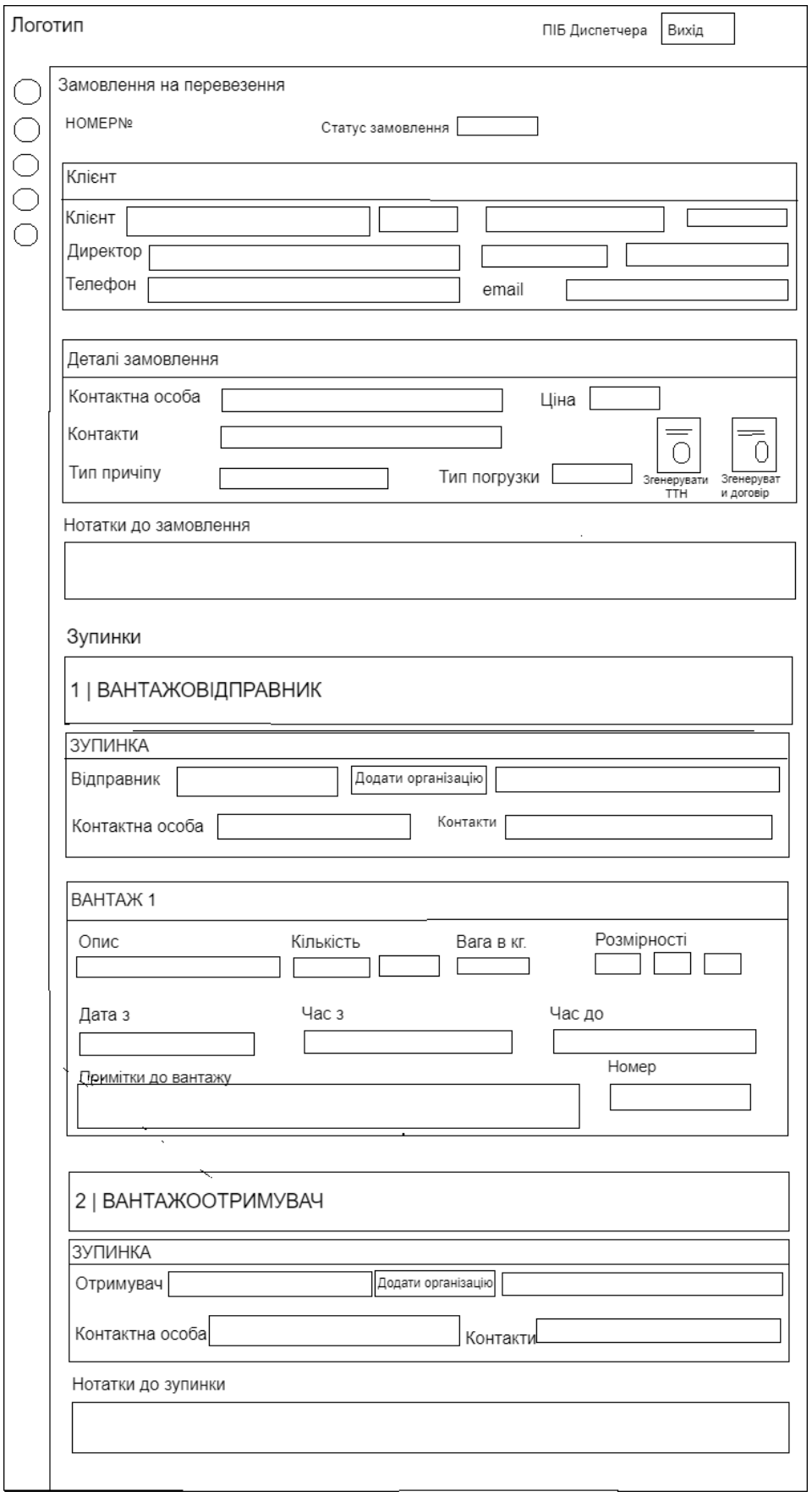


Рисунок 2.8 – Проект сторінки веб-застосунку для створення/редагування замовлення на перевезення

2.2 Технічний проект програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

Технічний проект – документація, яка містить остаточне технічне рішення і дає повне уявлення про будову програмного забезпечення, що проектується. Проаналізувавши ескізний проект були розроблені архітектура, статична та динамічна модель програмного забезпечення. Статична модель представлена діаграмою класів, а динамічна – діаграмою послідовності.

2.2.1 Архітектура програмного забезпечення для автоматизації роботи диспетчерського відділу

Веб-застосунок буде побудовано на класичній для клієнт-серверних веб-додатків трьох шаровій архітектурі «Модель-представлення-контролер». Цей шаблон передбачає поділ програми на три взаємопов’язані частини:

- модель – основний компонент архітектури, що керує даними та містить в собі бізнес-логіку програми;
- представлення – інтерфейс користувача для вводу та отримання даних;
- контролер – компонент, що передає запити від представлення до моделі та повертає відповідь від моделі до представлення.

Перевагами даної архітектури є більша впорядкованість і зрозумілість структури коду завдяки жорсткому поділу коду на компоненти. Також ці властивості допоможуть полегшити внесення змін та розширення програми в майбутньому.

На рисунку 2.9 зображена схема архітектури «Модель-представлення-контролер».

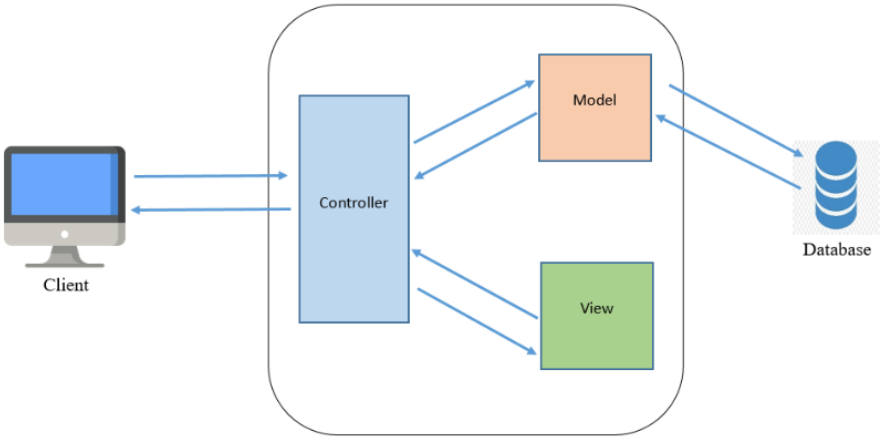


Рисунок 2.9 – Схема архітектури «Модель-представлення-контролер» [7]

2.2.2 Статична модель програмного забезпечення для автоматизації роботи диспетчерського відділу

Статична модель програмного забезпечення представляється діаграмою класів, яка демонструє загальну структуру ієрархії класів системи, їх кооперації, атрибути, методи, інтерфейси та взаємозв'язків між ними. Діаграма класів програмного представлена на рисунку 2.10.

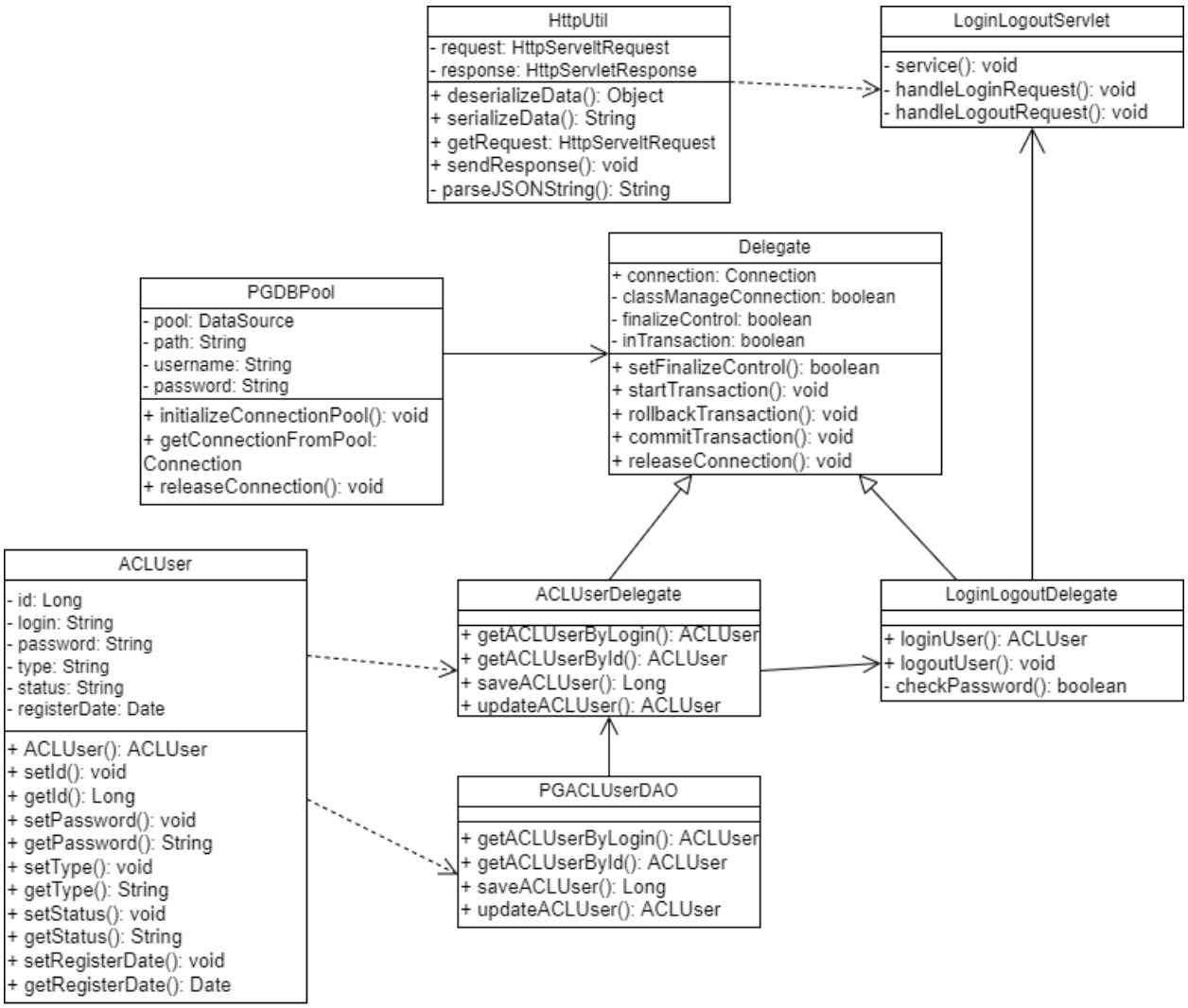


Рисунок 2.10 – Діаграма класів програмного забезпечення для автоматизації диспетчерського відділу логістичного підприємства «МістЕкспрес»

2.2.3 Специфікації класів програмного забезпечення для автоматизації роботи диспетчерського відділу

Для початку визначимо структурні елементи шаблону проектування «Модель-представлення-контролер»:

– контролер – прийом даних від представлення та їх передача до моделі та навпаки. В програмному забезпеченні, що розробляється, реалізується за допомогою класів, позначеними суфіксом «Servlet», наприклад «OrderServlet»;

– модель – основна бізнес-логіка програмного забезпечення. Реалізується за допомогою класів: з суфіксом «Delegate», наприклад «TripDelegate» – клас що має основну бізнес логіку пов’язану з обробкою Маршрутів на перевезення; з суфіксом «DAO», наприклад «TripDAO» – клас, що реалізує зв’язок з базою даних, займається операціями збереження та отримання даних з бази даних;

– представлення – реалізується за допомогою класичних фронт-енд технологій, тому в діаграмі класів не представлений.

Класи представлені на рисунку 2.10.

1) LoginLogoutServlet – клас, що представляє контролер. Цей клас використовує клас HTTPUtil для зв’язку з представленням через протокол HTTP.

Методи класу:

– service(): void – метод, що опрацьовує вхідні запити від клієнтської частини. Після того, як даний метод визначить, який саме прийшов запит він викличе один з нижчеописаних методів;

– handleLoginRequest(): void – метод, що опрацьовує запит на вхід користувача до системи;

– handleLogoutRequest(): void – метод, що опрацьовує запит на вихід користувача з системи.

2) HTTPUtil – допоміжний клас, який має методи для високорівневої взаємодії з HTTP протоколом.

Атрибути класу:

– request: HttpServletRequest – атрибут типу HttpServletRequest, що представляє собою об’єкт HTTP запиту;

– response: HttpServletResponse – атрибут типу HttpServletRequest, що представляє собою об’єкт HTTP відповіді.

Методи класу:

– deserializeData(): Object – метод, що перетворює дані, які прийшли з http запиту в Java клас;

– serializeData(): String – метод, що перетворює Java клас в дані, які призначені для надсилання відповіді на клієнтську частину;

– getRequest(): HttpServetRequest – метод для отримання об’єкту класу HttpServletRequest;

– sendResponse(): void – метод, що надсилає відповідь на клієнтську частину;

– parseJSONString(): String – метод, що перетворює java об’єкт в строку.

3) LoginLogoutDelegate – клас, що має в собі основну логіку прецеденту авторизації.

Методи класу:

– loginUser(): ACLUser – метод, що містить основну логіку авторизації користувача;

– checkPassword(): boolean – метод, що перевіряє правильність введення паролю;

– logoutUser(): void – метод, що містить основну логіку виходу користувача з системи.

4) ACLUser – клас, який є представленням сутності користувач. Атрибути класу є відображенням атрибутів сутності «Користувач» моделі даних. Клас відповідає принципу інкапсуляції, тому всі атрибути мають приватний модифікатор доступу, а доступ до атрибутів надається за допомогою «get» та «set» методів.

5) Delegate – клас, від якого унаслідуються усі класи делегати.

Атрибути класу:

– connection: Connection – об’єкт підключення до бази даних;

– classManageConnection: boolean – атрибут, що слугує для управління підключенням до бази даних;

– finalizeControl: boolean – атрибут, що слугує для управління підключенням до бази даних;

– inTransaction: boolean – атрибут, що слугує для управління з’єднанням до бази даних.

Методи класу:

– setFinalizeControl(): boolean – метод, що слугує для управління підключенням до бази даних;

– startTransaction(): void – метод, що розпочинає транзакцію;

– rollbackTransaction(): void – метод, що відміняє транзакцію;

– commitTransaction(): void – метод, що завершує транзакцію;

– releaseConnection(): void – метод, що віддає з'єднання до пулу з'єднань бази даних.

6) ACLUserDelegate – клас, що має в собі основну логіку для опрацювання даних сутності користувач.

Методи класу:

– getACLUserByLogin(): ACLUser – шукає клієнта в базі даних по його логіну та дістає його з бази даних;

– getACLUserById(): ACLUser – шукає клієнта в базі даних по його id та дістає його з бази даних;

– saveACLUser(): Long – зберігає клієнта в базі даних;

– updateACLUser(): ACLUser – оновлює клієнта в базі даних.

7) PGACLUserDAO – клас, який реалізує передачу даних сутності користувач між Java програмою та базою даних за допомогою SQL запитів.

– getACLUserByLogin(): ACLUser – реалізує пошук користувача в базі даних по його логіну та дістає його з бази даних;

– getACLUserById(): ACLUser – реалізує пошук користувача в базі даних по його id та дістає його з бази даних;

– saveACLUser(): Long – реалізує збереження користувача в базі даних;

– updateACLUser(): ACLUser – оновлює користувача в базі даних.

8) PGDBPool – клас, що створює та зберігає пул з'єднань до бази даних.

Атрибути класу:

– pool: DataSource – клас, що зберігає пул з'єднань до бази даних;

– path: String – шлях до бази даних;

– username: String – ім’я користувача, що підключається до бази даних;

– password: String – пароль користувача бази даних.

Методи класу:

– initializeConnectionPool(): void – метод, що відпрацьовує під час розгортання веб-застосунку та створює пул з’єднань до бази даних;

– getConnectionFromPool(): Connection – отримання з’єднання з пулу;

– releaseConnection(): void – покласти з’єднання в пул.

При кодуванні інших частин програмного забезпечення буде використаний такий самий шаблон проектування.

2.2.4 Динамічна модель програмного забезпечення для автоматизації роботи диспетчерського відділу

Динамічна модель реалізується за допомогою діаграми послідовності. На діаграмі послідовностей зображають об’єкти, які беруть учать у взаємодії, та саму взаємодію, але не відображають ніяких статичних зв’язків між об’єктами. Для діаграми послідовності ключовим моментом є динаміка взаємодії. Діаграма має як би два виміри. Перший зліва на право – відображає черговість передачі фокусу управління між об’єктами. Другий зверху вниз показує час життя об’єкта та час його активної діяльності.

Завдяки наявності осі часу діаграма послідовності краще визначає послідовність подій у процесі роботи програми. Тому діаграми послідовності є корисним засобом документування складних програмних систем.

На рисунку 2.11 зображена діаграма послідовності варіанту використання Д1 «Авторизація в системі».

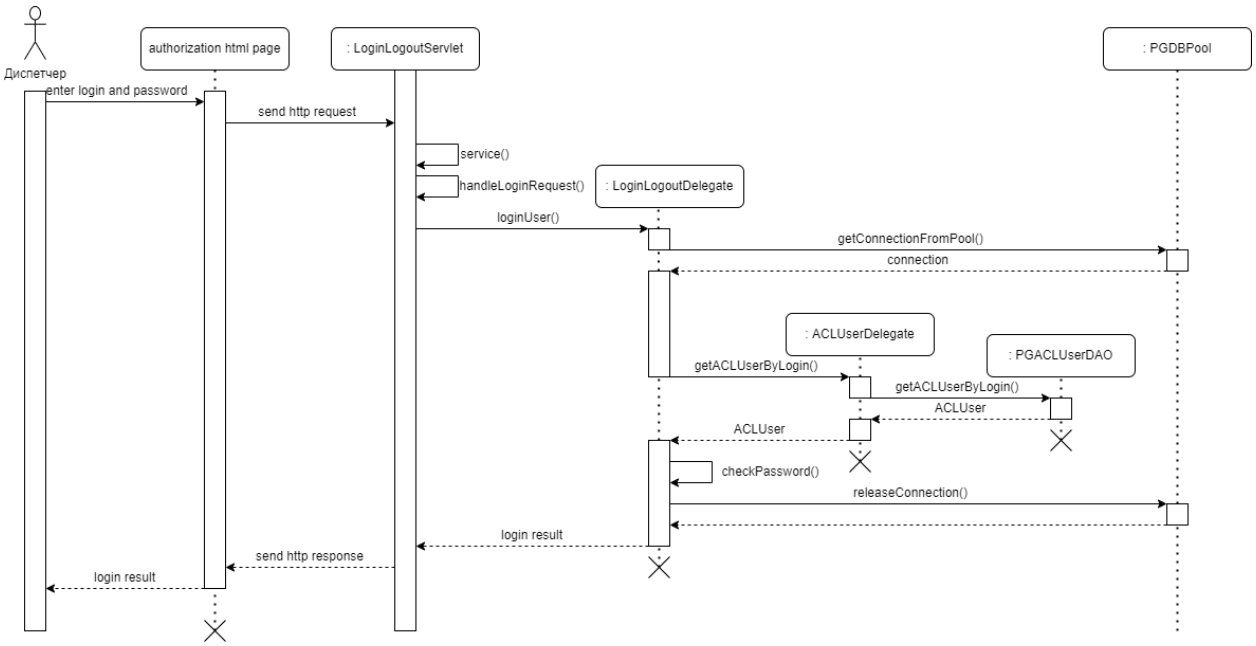


Рисунок 2.11 – Діаграма послідовності варіанту використання АДВ1 «Авторизація в системі»

2.2.5 Логічна модель даних програмного забезпечення для автоматизації роботи диспетчерського відділу

Логічна модель даних побудована на основі вище описаної концептуальної моделі даних. Але на відміну від концептуальної логічна модель даних описує дані для конкретної моделі даних (реляційна, об’єктна і т.п.). На цьому етапі не враховується деталі реалізації у конкретній СУБД, враховується лише специфіка обраної моделі даних.

Для зберігання даних веб-застосунку була обрана реляційна модель даних. База даних в ній представлена у вигляді набору відношень (relations). Відношення представляє з себе таблицю з заголовком, стовпцями і строками. Кожна строка являє собою групу взаємозв’язаних полів, які описують деякий об’єкт [8].

Логічна модель відображає набір відношень з первинними і зовнішніми ключами. Логічна модель даних веб-застосунку наведена на рисунку 2.12.

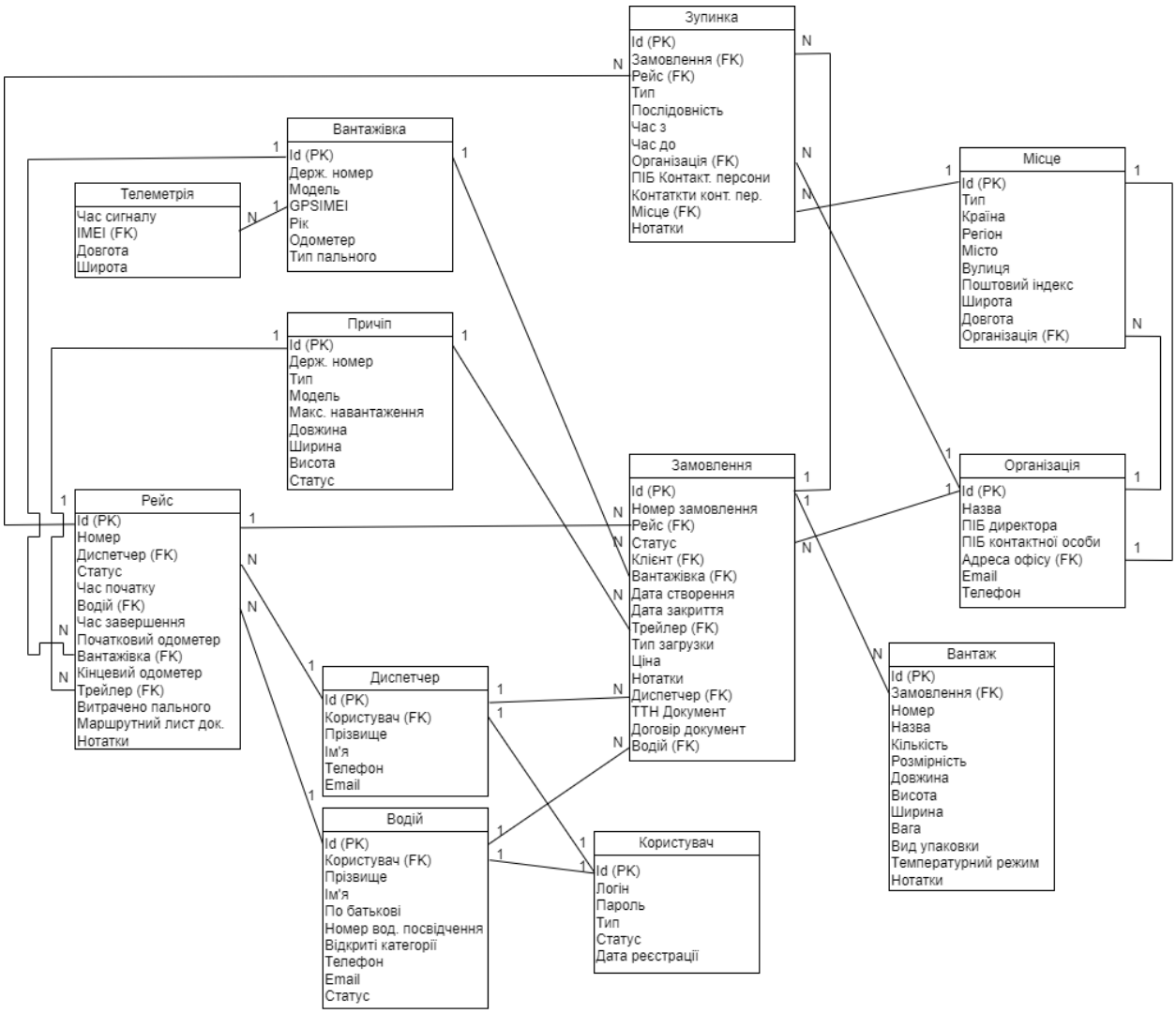


Рисунок 2.12 – Логічна модель даних веб-застосунку для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

2.3 Робочий проект програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

2.3.1 Обґрунтування вибору засобів програмування програмного забезпечення для автоматизації роботи диспетчерського відділу

Веб-застосунок, що розробляється, складається з серверної та клієнтської частин. Для розробки серверної частини була обрана корпоративна платформа Java Enterprise Edition. Це промислова технологія, яка в основному

використовується для високопродуктивних проектів, які потребують надійності, масштабованості та гнучкості. Платформа надає API та виконавче середовище для розробки і виконання корпоративного програмного забезпечення, включаючи мережеві та веб сервіси, та інші масштабовані, розподілені додатки. Java EE розширює стандартну платформу Java (Java SE - Java Standard Edition) [9].

Java – об'єктно-орієнтована мова зі строгою типізацією, що має C-подібний синтаксис. Мова Java реалізує принцип "Write once, run everywhere" – програми написані на Java можна виконати на будь-якій платформі, де встановлено JRE (Java Runtime Environment) – середовище виконання Java. Програми, написані на Java, перетворюються в байт-код, який виконує JVM (Java Virtual Machine), яка є частиною JRE, та яка не залежить від апаратної платформи. Вбудований в JRE механізм управління оперативною пам'яттю (garbage collector) дозволяє звільнити розробників від ручного вивільнення оперативної пам'яті, тим самим дозволяє зменшити кількість помилок під час написання програм.

Можна виділити наступні переваги використання мови Java та платформи Java Enterprise Edition:

- простота синтаксису;
- об'єктно-орієнтовність дозволяє створювати модульні програми та багаторазово перевикористовувати код;
- платформонезалежність дозволить виконувати Java програму на сервері з будь-якою встановленою операційною системою;
- java стабільна мова з повною зворотньою сумісністю;
- java поширена мова програмування, тому в процесі розробки не з'явиться питань на які немає відповідей;
- великий вибір бібліотек та API для вирішення широкого спектру задач.

Для розробки клієнтської частини був обраний JavaSript фреймворк AngularJS, який призначений для розробки односторінкових веб-застосунків, що складаються з однієї HTML сторінки, CSS і JavaScript. AngularJS широко

використовується для створення великомасштабних систем, які задовольняють потреби підприємств або організацій, а не потреби конкретного користувача.

Можна виділити наступні переваги використання AngularJS:

– модульна структура дозволяє ефективно поділяти код на модулі, що допомагає ефективно організовувати функціонал та створювати взаємозамінні модулі, що скорочує час на розробку;

– великий вибір готових рішень, що спростить та пришвидшить розробку веб-сторінок та скриптів до них.

Для збереження даних була обрана об’єктно реляційна система управління базами даних PostgreSQL. Основні переваги використання PostgreSQL при розробці корпоративних веб-застосунків:

– потужні та надійні механізми транзакцій та реплікації;

– безкоштовна СУБД;

– наявність модулю PostGIS для роботи з геометричними типами даних, що актуально для логістичного підприємства, яке використовує ці дані для відображення площин на мапі.

PostgreSQL у зв’язці платформою JavaEE є найкращим варіантом для створення корпоративних веб-застосунків, особливо за умови зберігання великих об’ємів інформації у базі.

2.3.2 Фізична модель даних програмного забезпечення для автоматизації роботи диспетчерського відділу

Фізична модель даних створюється для подальшої реалізації на конкретній СУБД, при проектуванні даного програмного забезпечення використовується СУБД PostgreSQL. На відміну від попередньої логічної моделі даних, фізична модель даних включатиме всі артефакти обраної бази даних, які необхідні для створення зв’язків між таблицями. Ціль побудови фізичної моделі даних – перехід від формальних атрибутів, які

використовувалися на етапі проектування, до фактичних атрибутів, які будуть використані при розробці бази даних.

Фізичну модель даних наведено на рисунку 2.13.

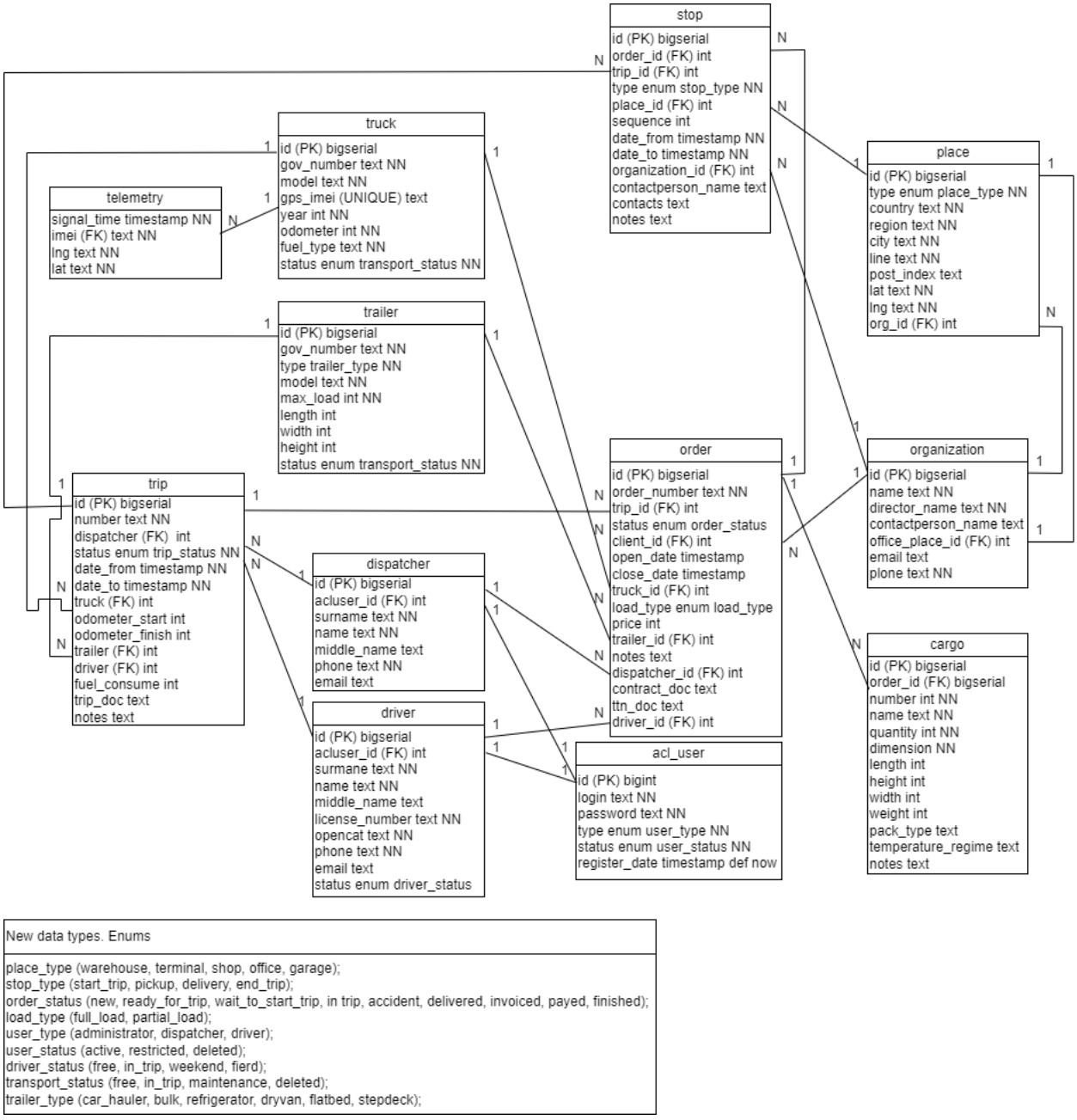


Рисунок 2.13 – Фізична модель даних програмного забезпечення для автоматизації роботи диспетчерського відділу

2.3.3 Діаграма компонентів програмного забезпечення для автоматизації роботи диспетчерського відділу

Діаграма компонентів описує особливості фізичного представлення системи. Діаграма компонентів програмного забезпечення для автоматизації роботи диспетчерського відділу представлена на рисунку 2.14.

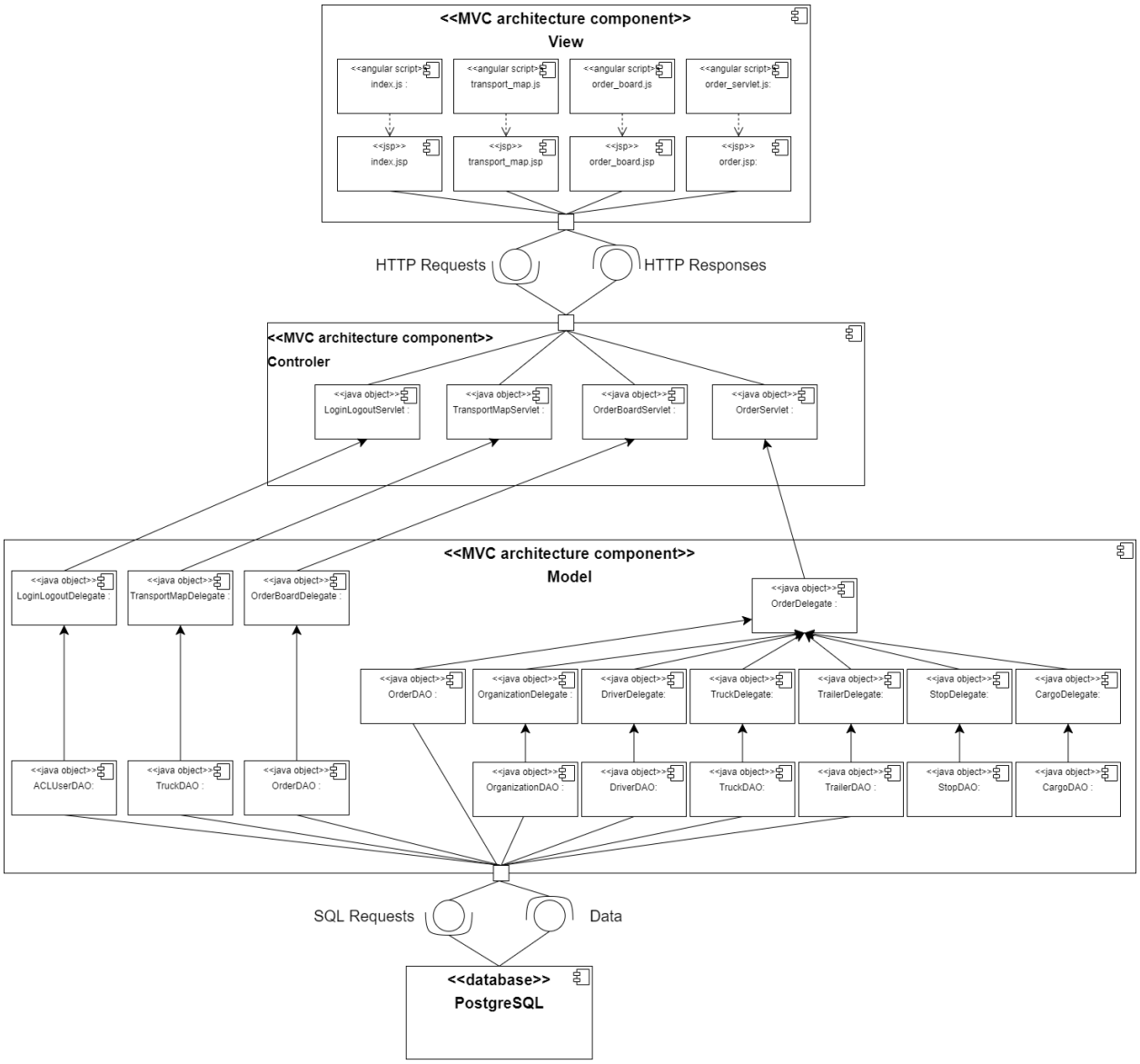


Рисунок 2.14 – Діаграма компонентів програмного забезпечення

Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється, встановивши відносини та залежності між програмними компонентами, у ролі яких можуть виступати файли програми, класи, бази

даних, інтерфейс користувача або пристрої. У багатьох середовищах розробки модуль чи компонент відповідає файлу. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси та залежності між ними [10].

2.3.4 Діаграма розгортання програмного забезпечення для автоматизації роботи диспетчерського відділу

Діаграма розгортання – діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах [6].

Веб застосунок, що проектується буде розгортатися на двох серверах. Перший з яких буде сервером бази даних і буде містити в собі встановлену СУБД PostgreSQL. Другий сервер буде бек-енд сервером, що міститиме Java програму, яка реалізує бізнес логіку, зв’язок з клієнтом, користувацький інтерфейс, зв’язок з сервером бази даних. На другому сервері буде встановлена Java Virtual Machine, яка буде виконувати контейнер сервлетів Tomcat, в якому буде міститися вихідний код Java програми. Клієнт буде взаємодіяти з програмним забезпеченням за допомогою персонального комп’ютера, або смартфону зі встановленим браузером. Зв’язок між клієнтом та двома серверами буде відбуватися через мережу Інтернет.

Діаграму розгортання веб-застосунку для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес» наведено на рисунку 2.15.

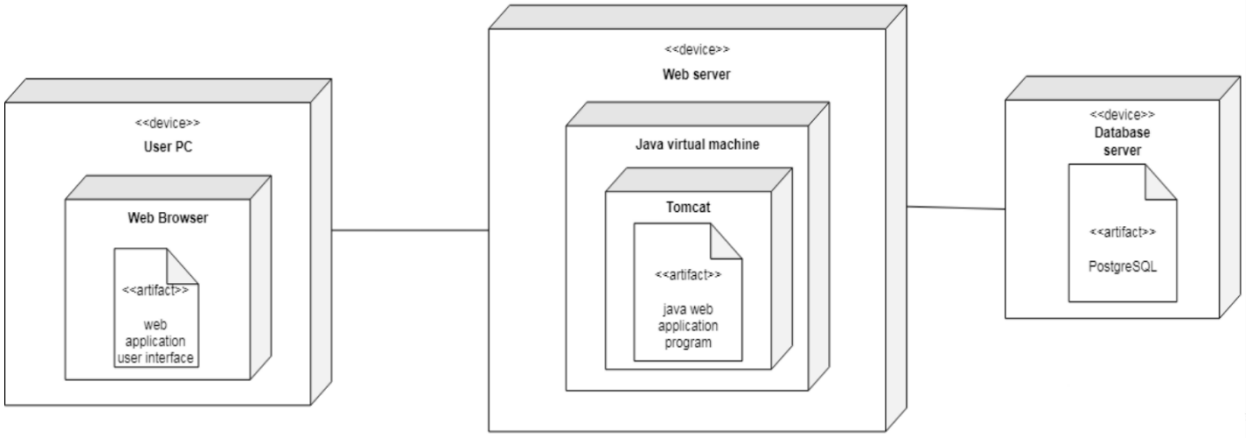


Рисунок 2.15 – Діаграма розгортання програмного забезпечення для автоматизації диспетчерського відділу логістичного підприємства «МістЕкспрес»

2.3.5 Реалізація основних класів програмного забезпечення для автоматизації роботи диспетчерського відділу

Програмне забезпечення пишеться за допомогою технологій Java EE та AngularJS. Нижче буде представлений код програми, що представляє початковий запуск серверу та відображення головної сторінки – авторизації. Класи програмного забезпечення будуть завантажені в контейнер сервлетів Tomcat, який за допомогою файлу web.xml з URL адресами буде визначати, який саме сервлет буде опрацьовувати запит від клієнта. Під час завантаження проекту спочатку відпрацьовує метод contextInitializer() класу AppStartup, який ініціалізує контекст виконання програмного забезпечення. Після цього за адресою серверу автоматично стане доступним файл index.jsp та index.js, які створюють користувацький інтерфейс для авторизації. Клас LoginLogoutServlet буде контролером для сторінки авторизації. Клас PGDBPool слугує для взяття та вивільнення з’єднання з базою даних.

web.xml

```
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >
```

```
<web-app>

    <display-name>Archetype Created Web Application</display-name>

    <context-param>
        <param-name>APP_LINK</param-name>
        <param-value>http://localhost:8080/app</param-value>
    </context-param>

    <resource-ref>
        <res-ref-name>jdbc/app</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>

    <listener>
        <listener-class>app.AppStartup</listener-class>
    </listener>

    <error-page>
        <error-code>401</error-code>
        <location>/html/error/401.html</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/html/error/404.html</location>
    </error-page>

    <servlet>
        <servlet-name>LoginLogoutServlet</servlet-name>
        <servlet-class>app.servlet.LoginLogoutServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginLogoutServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>ACLUserServlet</servlet-name>
        <servlet-class>app.servlet.ACLUserServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ACLUserServlet</servlet-name>
        <url-pattern>/user</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>OrganizationServlet</servlet-name>
        <servlet-class>app.servlet.OrganizationServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>OrganizationServlet</servlet-name>
        <url-pattern>/organization</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>OrderServlet</servlet-name>
        <servlet-class>app.servlet.OrderServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>OrderServlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
```

```
    <servlet>
        <servlet-name>DriverServlet</servlet-name>
        <servlet-class>app.servlet.DriverServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DriverServlet</servlet-name>
        <url-pattern>/driver</url-pattern>
    </servlet-mapping>

</web-app>
```

AppStartup.java

```
package app;
import javax.naming.NamingException;
import java.io.IOException;
import java.util.logging.Logger;


public class AppStartup implements ServletContextListener {
    private static final Logger log = Logger.getLogger(AppStartup.class);
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        try {
            ServletConstants.APP_LINK =
sce.getServletContext().getInitParameter("APP_LINK");
            AppConstants.initializeProperties();
            PGDBPool.initializeConnectionPool();
            ArgonInitialize.getInstance();
        } catch (IOException | NamingException e) {
            log.error("Cannot initialize application", e);
        }
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {}
}
```

index.jsp

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="libs/bootstrap-4.5.2-
dist/css/bootstrap.min.css">
    <!-- Popper JS -->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.j
s"></script>
    <!-- jQuery library -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scri
pt>
    <!-- Latest compiled JavaScript -->
    <script src="libs/bootstrap-4.5.2-dist/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/main-style.css">
    <link rel="stylesheet" href="css/article.css">
```

```
    <script src="libs/angular-1.8.0/angular-animate.js"></script>
    <script src="js/appModule.module.js"></script>
    <script src="js/appConstants.component.js"></script>
    <script src="js/index.component.js"></script>

</head>

<!-- Modal log -->
<div id="logModal" class="modal fade" role="dialog">
    <div class="modal-dialog">

        <!-- Modal content-->
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">Вхід</h4>
                <button type="button" class="close" data-
dismiss="modal">&times;</button>
            </div>

            <form class="form" name="loginForm">
                <div class="modal-body">
                    <div class="form-group">
                        <input name="name" ng-model="vm.loginUser.login"
type="email" class="form-control"  placeholder="email" required>
                    </div>
                    <div class="form-group">
                        <input name="password" ng-
model="vm.loginUser.password" type="password" class="form-control"
placeholder="пароль" required>
                    </div>
                </div>

                <div class="modal-footer">
                    <button type="submit" ng-click="vm.loginUser(loginForm)"
name="submit" class="btn btn-submit" value="submit">Увійти</button>
                </div>

            </form>
        </div>
    </div>
</div>

</html>
```

index.js

```
(function () {
    'use strict';

    angular
        .module('appModule')
        .controller('indexController', Controller);

    Controller.$inject = ['$http', '$cookies', '$uibModal', 'APP_LINK'];

    function Controller($http, $cookies, $uibModal, APP_LINK) {
        const vm = this;

        vm.isLogin = false;
        vm.currentUser = {};

        vm.loginUser = {
            login: null,
```

```
            password: null
        };

        vm.loginUser = loginUser;
        vm.invalidateSession = invalidateSession;

        activate();

        function activate() {
            getLoginUser();
        }

        function getLoginUser() {
            $http.post(APP_LINK + 'app/user?action=getUserById')
                .then((response => {
                    console.log('RESPONSE:', response);
                    vm.isLogin = true;
                    vm.currentUser = response.data;
                }), (() => {}));
        }

        function loginUser(form) {
            if (!form.$valid) return;
            $http.post(APP_LINK +
'login?action=login&login='+vm.loginUser.login+'&password='+vm.loginUser.pass
word)
                .then((response => {
                    console.log('RESPONSE:', response);
                    window.location.href = "html/incomeBook.component.html";
                }), (
                    (response) => {
                        if (response.status === 401) {
                            alert('Невірний логін або пароль');
                        } else {
                            alert('Щось пішло не так!');
                        }
                    }));
        }

        function invalidateSession() {
            $http.post(APP_LINK + 'login?action=logout')
                .then(
                    () => {
                        window.location.href = "index.jsp"; //This line
redirects (redirect in Java does not work? because this method $http.get()
асинхроннij).
                    },
                    () => {
                        alert('Виникла помилка. Повідомте будь ласка
розробників.');
                    }
                );
        }

    }
})();
```

LoginLogoutServlet.java

```
package app.servlet;
import app.HttpUtil;
import app.delegate.LoginLogoutDelegate;
import app.vo.ACLUser;
```

```
import java.io.UnsupportedEncodingException;
import java.util.logging.Logger;

public class LoginLogoutServlet extends HttpServlet {
    private static final Logger log =
Logger.getLogger(LoginLogoutServlet.class);
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
{
        HttpUtil su = new HttpUtil(req, resp);
        try {
            su.getRequest().setCharacterEncoding("UTF-8");
        } catch (UnsupportedEncodingException e) {
            log.error(e);
        }
        String action = req.getParameter(ServletConstants.ACTION);
        switch (action) {
            case ServletConstants.LOGIN : handleLoginRequest(su); break;
            case ServletConstants.LOGOUT : handleLogoutRequest(su); break;
            default: su.sendResponse(500, "Server error");
        }
    }

    private void handleLoginRequest(HttpUtil su) {
        try {
            String login = su.getRequest().getParameter("login");
            String password = su.getRequest().getParameter("password");

            LoginLogoutDelegate loginLogoutDelegate = new
LoginLogoutDelegate();
            ACLUser aclUser = loginLogoutDelegate.loginUser(login, password);

            if (aclUser == null) {
                log.info("Access denied for: login name = " + login + " ,
password = " + password);
                su.sendResponse(401, "Неправильний логін або пароль");
                return;
            }

            HttpSession session = su.getRequest().getSession(true);
            session.setAttribute(ServletConstants.ATTRIBUTE_NAME_USER_ID,
aclUser.getId());

            Cookie languageCookie = new
Cookie(ServletConstants.COOKIE_NAME_LANGUAGE, language);
            languageCookie.setMaxAge(-1); // до кінця сесії
            su.getResponse().addCookie(languageCookie);

            log.info("User logged in: login name = " + login + " , password =
" + password);
            su.getResponse().sendRedirect(ServletConstants.APP_LINK +
"app/transport-map");
        } catch (Exception e) {
            log.error("SERVLET LOGIN Cannot login session. actionLogin()",
e);
            su.sendError(e.getMessage());
        }
    }

    private void handleLogoutRequest(HttpUtil su) {
        try {
            LoginLogoutDelegate userDelegate = new LoginLogoutDelegate();
            userDelegate.logoutUser(su);
```

```
        } catch (Exception e) {
            log.error("SERVLET LOGOUT Cannot logout session. actionLogout()",
e);
            su.sendError(e.getMessage());
        }
    }

}
```

PGDBPool.java

```
package app;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.logging.Logger;

public class PGDBPool {
    private static final Logger log = Logger.getLogger(PGDBPool.class);
    private static DataSource pool;
    private static String path = "java:comp/env/jdbc/dispapp";
    private static String username = "vladdr";
    private static String password = "511770";


    public static void initializeConnectionPool() throws NamingException {
        InitialContext initContext = new InitialContext();
        pool = (DataSource) initContext.lookup(path, username, password);
    }

    public static Connection getConnectionFromPool() throws SQLException {
        try {
            return pool.getConnection();
        } catch (SQLException e) {
            log.error("PGDBPool Cannot get connection from pool", e);
            throw e;
        }
    }

    public static void releaseConnection(Connection connection) {
        try {
            if (connection != null) connection.close();
        } catch (SQLException e) {
            log.error("PGDBPool Cannot release connection", e);
        }
    }
}
```

2.3.5 Реалізація користувацького інтерфейсу програмного забезпечення для автоматизації роботи диспетчерського відділу

На рисунках 2.16, 2.17, 2.18, 2.19 зображені розроблені користувацькі інтерфейси програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

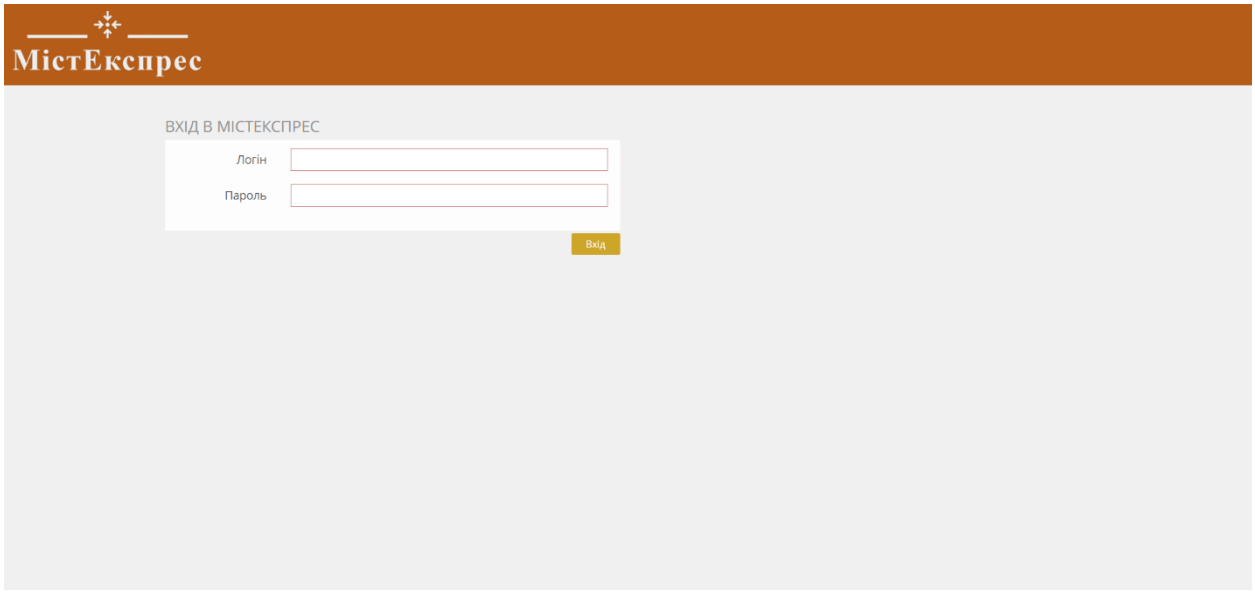


Рисунок 2.16 – Сторінка авторизації

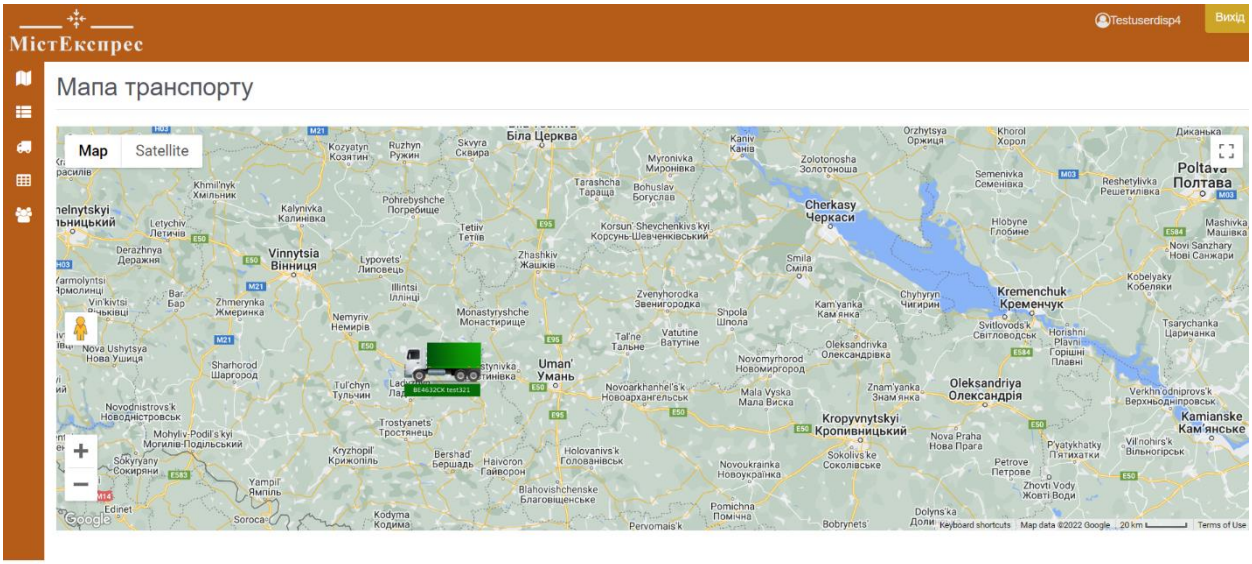
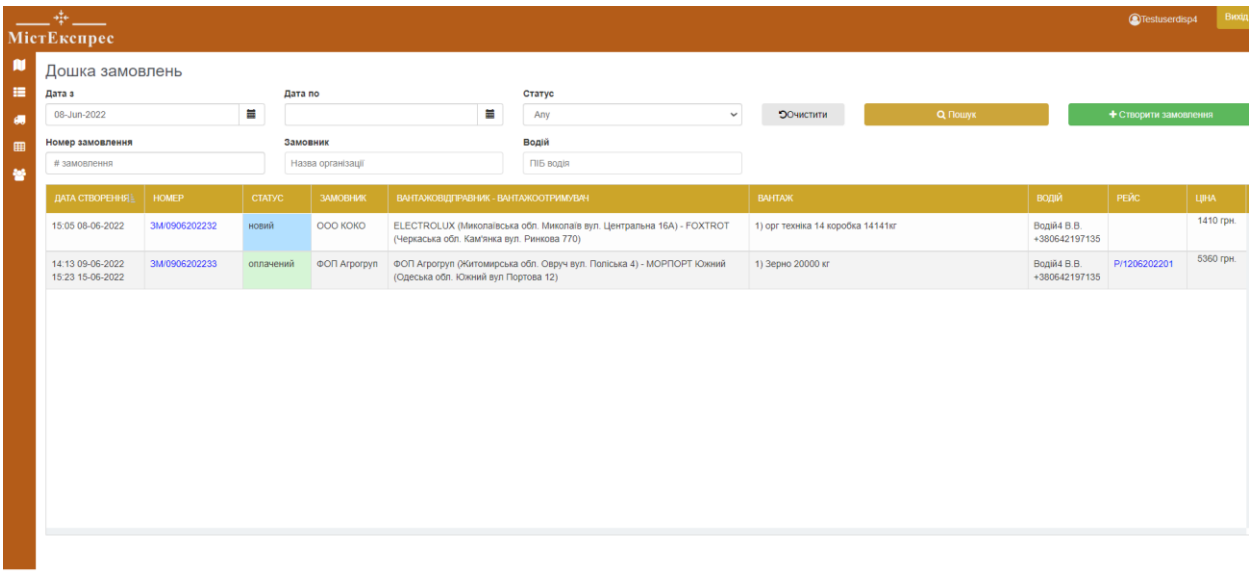


Рисунок 2.17 – Сторінка з мапою транспорту



Сторінка 2.18 – Сторінка з дошкою замовлень на перевезення

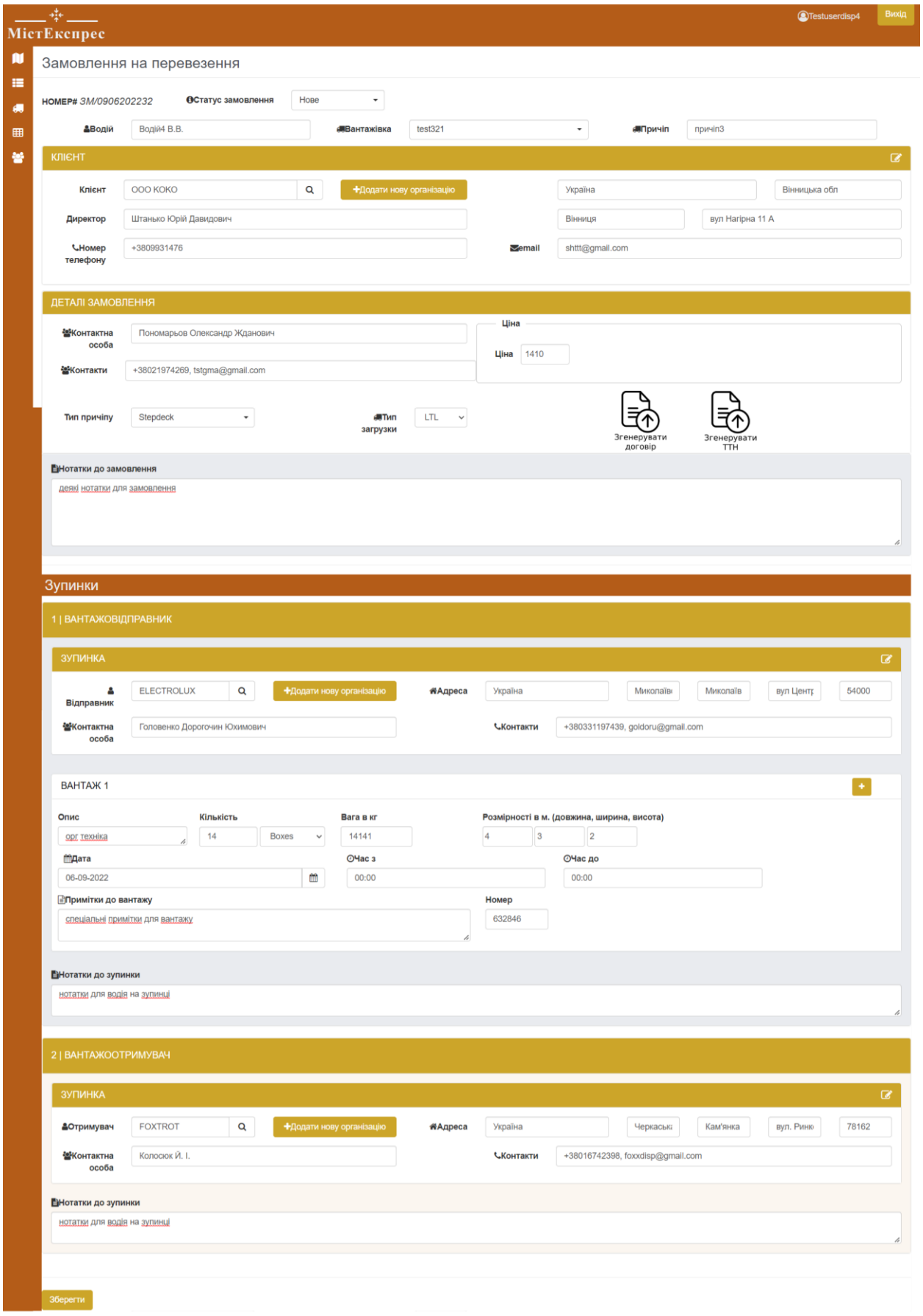


Рисунок 2.19 – Сторінка для створення/редагування замовлення

2.3.6 Тестування програмного забезпечення для автоматизації роботи диспетчерського відділу

Під час тестування програмного забезпечення проводиться випробування програмного продукту, що має на меті перевірку відповідності між реальною поведінкою програми та її очікуваною поведінкою на кінцевому наборі тестів, вибраних певним чином [11].

Для проведення тестування було обрано два варіанти використання та для кожного з них був обраний найбільш ефективний метод розробки тестових випадків при тестуванні «чорної скриньки».

Тестування варіанту використання АДВ1 «Авторизація в системі» за допомогою методу тестування еквівалентного розбиття

В таблиці 2.6 виділені усі можливі правильні та неправильні класи еквівалентності для вхідних даних, в даному випадку це логін та пароль.

Таблиця 2.6 – Класи еквівалентності вхідних даних

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
логін	1) строка, яка відповідає шаблону email	2) порожня строка 3) строка, що не відповідає шаблону email
пароль	4) строка довжиною від 8 до 16 символів включно, що не містить символів лапок, пробілу та двокрапки	5) порожня строка 6) строка довжиною від 1 до 7 символів, або від 17 і більше 7) строка, що містить символи лапок, пробілу та двокрапки

Тести для правильних класів еквівалентності наведені в таблиці 2.7.

Таблиця 2.7 – Тестування правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні дані	Результат проходження тесту
1	1, 4	логін: vladdisp1@gmail.com пароль: 89Jgp!770D_!	запит на сервер для перевірки логіну та паролю	пройдений

Комбінації кожного правильного класу з рештою неправильних класів еквівалентності наведені в таблиці 2.8.

Таблиця 2.8 – Тестування комбінації кожного правильного класу з рештою неправильних класів

№ тесту	Покритий клас	Вхідні дані	Правильні вихідні дані	Результат проходження тесту
1	1, 5	логін: vladdisp1@gmail.com пароль:	повідомлення «поле пароль обов’язкове для заповнення»	пройдений
2	1, 6	логін: vladdisp1@gmail.com пароль: 89sa	повідомлення «пароль повинен мати довжину від 8 до 16 символів»	пройдений
3	1, 7	логін: vladdisp1@gmail.com пароль: 89SD”wfq353	повідомлення «пароль не повинен містити символів лапок, пробілу та двокрапки»	пройдений
4	4, 2	логін: пароль: 89Jgp!770D_!	повідомлення «поле логін обов’язкове до заповнення»	пройдений
5	4, 3	логін: vladdisp1 пароль: 89Jgp!770D_!	повідомлення «логін повинен бути вашим email»	пройдений

Тестування варіанту використання ДЗ «Створення/редагування замовлення на перевезення» за допомогою методу тестування варіантів використання

Основна ідея даного методу полягає в тому, що кожен варіант використання може складатися з кількох різних сценаріїв поведінки, і кожен окремий сценарій є потенційним тестовим випадком, тому що кожен сценарій виявляє набір відповідей системи на зовнішні впливи.

Сценарій 1.

Передумова: диспетчер узгодив деталі замовлення на перевезення з клієнтом.

Основний потік подій:

1. Вибрати клієнта зі списку існуючих організацій
2. Заповнити розділ «Деталі замовлення» тестовими даними
3. Вибрати вантажовідправника зі списку існуючих організацій
4. Вибрати адрес вантажовідправника зі списку існуючих адрес
5. Заповнити розділ «Вантаж» тестовими даними
6. Вибрати вантажоотримувача зі списку існуючих організацій
7. Вибрати адрес вантажоотримувача зі списку існуючих адрес
8. Обрати вільного водія і вантажівку та підходящий для транспортування причіп
9. Зберегти замовлення на перевезення.

Очікувані результати:

1. Замовлення на перевезення збережеться у списку замовлень зі статусом «Нове»
2. На дошці зайнятості водіїв буде відмічений відрізок часу протягом якого водій буде виконувати новостворене замовлення на перевезення
3. На дошці зайнятості вантажівок буде відмічений відрізок часу протягом якого вантажівка буде використовуватись для виконання замовлення

4. На дошці зайнятості причепів буде відмічений відрізок часу протягом якого причіп буде використовуватись для виконання замовлення.

Сценарій 2

Передумова: диспетчер узгодив деталі замовлення з клієнтом.

Основний потік подій:

1. Спробувати знайти клієнта в списку існуючих організацій
2. Створити нову організацію та обрати її в якості клієнта для замовлення на перевезення
3. Заповнити розділ «Деталі замовлення» тестовими даними
4. Спробувати вибрати вантажовідправника зі списку існуючих організацій
5. Створити нову організацію та місце для неї
6. Обрати новостворену організацію в якості вантажовідправника
7. Вибрати новостворений адрес вантажовідправника зі списку існуючих
8. Заповнити розділ «Вантаж» тестовими даними
9. Спробувати вибрати вантажоотримувача зі списку існуючих організацій
10. Створити нову організацію та місце для неї
11. Обрати новостворену організацію в якості вантажоотримувача
12. Вибрати новостворений адрес вантажоотримувача зі списку існуючих
13. Обрати вільного водія і вантажівку та підходящий для транспортування причіп
14. Зберегти замовлення на перевезення.

Очікувані результати:

1. Замовлення на перевезення збережеться у списку замовлень зі статусом «Нове»

2. На дошці зайнятості водіїв буде відмічений відрізок часу протягом якого водій буде виконувати новостворене замовлення на перевезення

3. На дошці зайнятості вантажівок буде відмічений відрізок часу протягом якого вантажівка буде використовуватись для виконання замовлення

4. На дошці зайнятості причепів буде відмічений відрізок часу протягом якого причіп буде використовуватись для виконання замовлення.

5. В списку організацій з’являться три новостворені організації

6. Для кожної новоствореної організації з’являться по два місця.

Сценарій 3

Передумова: диспетчер узгодив деталі замовлення на перевезення з клієнтом.

Основний потік подій:

1. Вибрати клієнта зі списку існуючих організацій

2. Заповнити розділ «Деталі замовлення» тестовими даними

3. Вибрати вантажовідправника зі списку існуючих організацій

4. Вибрати адрес вантажовідправника зі списку існуючих адрес

5. Заповнити розділ «Вантаж» тестовими даними

6. Вибрати вантажоотримувача зі списку існуючих організацій

7. Вибрати адрес вантажоотримувача зі списку існуючих адрес

8. Обрати вільного водія і вантажівку та не підходящий для транспортування причіп

9. Зберегти замовлення на перевезення.

Очікувані результати:

1. При збереженні замовлення на перевезення диспетчер отримає попередження, що габарити причіпу не відповідають вантажу, що має перевозитись

2. Замовлення на перевезення не збережеться.

В таблиці 2.9 наведений результат тестування варіанту використання «Створення/редагування замовлення на перевезення» за допомогою методу тестування варіантів використання.

Таблиця 2.9 – Результат тестування

Сценарій	Результат проходження тесту
1	пройдений
2	пройдений
3	пройдений

2.3.7 Випробування програмного забезпечення для автоматизації роботи диспетчерського відділу

Випробування програмного забезпечення проводиться згідно програми та методики випробувань, яка наведена в додатку Б.

Були перевірені наступні функції: авторизація користувача, контроль руху вантажівок, створення замовлення на перевезення, генерація супровідної документації, створення рейсу, перевірка інформативності дошки зайнятості, створення організації, створення вантажівки, реєстрація користувача, редагування водієм поточного рейсу.

Авторизація

Перейшовши на сторінку авторизації було введено логін: vladd@gmail.com та пароль: qw89770!LR7. Після натискання кнопки «Увійти» логін та пароль був перевірений на сервері та у відповідь було виконано перенаправлення на головну сторінку диспетчера, що відповідає очікуваному результату.

Контроль руху вантажівок

Авторизувавшись як диспетчер перейшли на сторінку з мапою транспорту. Виявили, що увесь транспорт підприємства відображається на мапі транспорту, що відповідає очікуваному результату.

Створення замовлення на перевезення

Авторизувавшись як диспетчер перейшли на сторінку створення/редагування замовлення. Правильно заповнили всі поля, та натиснули на кнопку «Зберегти». В результаті замовлення було збережене та згодом відображене у списку замовлень, що відповідає очікуваному результату.

Генерація супровідної документація

Авторизувавшись як диспетчер перейшли на сторінку створення/редагування вже створеного замовлення та натиснули кнопки «Згенерувати ТТН» та «Згенерувати договір на перевезення». В результаті було згенеровано за збережено два документи на основі вже збереженого замовлення на перевезення, шо відповідає очікуваному результату.

Створення рейсу

Авторизувавшись як диспетчер перейшли на сторінку створення/редагування рейсу та обрали два замовлення на основі яких буде формуватися рейс. Заповнивши усі необхідні поля натиснули на кнопку «Зберегти». В результаті рейс був збережений та згодом відображений у списку рейсів, що відповідає очікуваному результату.

Перевірка інформативності дошки зайнятості

Авторизувавшись як диспетчер перейшли на сторінку дошки зайнятості. Було виявлено, що інформація про час зайнятості водіїв, вантажівок, причепів відображається правильно, що відповідає очікуваному результату.

Створення організації

Авторизувавшись як диспетчер перейшли на сторінку створення/редагування організації. Заповнивши необхідні поля для створення організації натиснули на кнопку «Зберегти». В результаті організація була збережена та згодом відображена у списку організацій, що відповідає очікуваному результату.

Створення вантажівки

Авторизувавшись як адміністратор перейшли на сторінку створення/редагування активів підприємства та обрали вантажівку в якості об’єкта створення. Заповнивши всі поля натиснули на кнопку «Зберегти». В результаті вантажівка була збережена та згодом відображена у списку вантажівок підприємства, що відповідає очікуваному результату.

Реєстрація користувача

Авторизувавшись як адміністратор перейшли на сторінку створення/редагування користувача та обравши водія в якості об’єкта створення. Заповнивши всі поля натиснули на кнопку «Зберегти». В результаті водій був збережений та згодом відображений у списку водіїв підприємства, що відповідає очікуваному результату.

Редагування водієм поточного рейсу

Авторизувавшись як водій перейшли на сторінку створення/редагування поточного рейсу. Заповнивши необхідні поля натиснули на кнопку «Зберегти». В результаті відредагований рейс був збережений, що відповідає очікуваному результату.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

За результатами кваліфікаційної роботи було розроблено програмне забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес». Процес розробки почався з аналізу організаційної структури підприємства та аналізу предметної галузі, що дозволило виявити цілі та задачі, які постають перед програмним забезпеченням. Аналіз існуючих програм-аналогів дозволив зрозуміти, яких функцій не вистачає програмному забезпеченню, що використовується в області вантажоперевезень, та чим саме програмне забезпечення, що розроблюється, буде перевершувати вже існуюче. За результатами аналізу предметної галузі та програм аналогів була обґрунтована доцільність розробки та поставлена задача на розробку програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

Під час постановки задачі були виділені вимоги до програмного забезпечення, які згодом були формалізовані при побудові моделі варіантів використання. Для основних варіантів використання була представлена специфікація за допомогою діаграм діяльності.

За результатом моделювання вимог були побудовані концептуальна, логічна та фізична моделі даних.

Був створений проект графічного інтерфейсу користувача, на основі якого потім був побудований кінцевий варіант графічного інтерфейсу.

Була визначена архітектура побудови програмного забезпечення. На основі архітектури була побудована діаграма компонентів та статична модель програмного забезпечення за допомогою діаграми класів, що представляла собою загальну структуру ієрархії класів в системі. Для кожного класу були

описані його специфікації. Для визначення послідовності подій у процесі роботи програми була побудована динамічна модель програмного забезпечення за допомогою діаграми послідовностей.

На наступному етапі були обрані засоби розробки, такі як мова програмування та СУБД. В якості СУБД була обрана PostgreSQL, в якості мов програмування були обрані технології Java EE та AngularJS. Для пояснення, того, як програмне забезпечення буде розгортатися на сервері, була побудована діаграма розгортання. На обраних засобах розробки було проведено кодування програмного забезпечення.

Після завершення кодування було проведено тестування основних варіантів використання за допомогою методів еквівалентного розбиття та тестування варіантів використання.

Останнім етапом було проведення випробувань за розробленою методикою. Тестування та випробування показало, що розроблене програмне забезпечення реалізує вимоги, що були для нього виставленими на етапі визначення вимог.

4 ОХОРОНА ПРАЦІ

Процес розробки програмного забезпечення завжди має відбуватись в умовах нанесення мінімальної шкоди здоров'ю працівника. Метою дипломної роботи є розробка програмного забезпечення для диспетчерської логістичного підприємства, що означає виконання за персональним комп'ютером великого обсягу роботи протягом тривалого часу, тому доцільно нагадати важливість дотримання норм охорони праці та техніки безпеки під час роботи з комп'ютером. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Цей нормативний документ визначає вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та персональної електронно-обчислювальної машини (ПЕОМ), гігієнічні вимоги до організації і обладнання робочих місць, вимоги до режимів праці і відпочинку, вимоги до профілактичних медичних оглядів, гігієнічні вимоги до параметрів виробничого середовища приміщень. За допомогою цих стандартів можна досягти значного зниження наслідків несприятливої дії шкідливих та небезпечних факторів на програмістів.

4.1 Вимоги до виробничого приміщення

Відповідно до встановлених гігієнічно-санітарних вимог (ДСТ 12.1.005-88, СН 4088-86) роботодавець зобов'язаний забезпечити в приміщеннях з ВДТ оптимальні параметри виробничого середовища.

Природне освітлення в приміщеннях з ВДТ має здійснюватися через вікна, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні

екранів ПК і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Основні вимоги до виробничого приміщення для експлуатації ВДТ:

— воно не може бути розміщено у підвалах та цокольних поверхах;

— площа на одне робоче місце в такому приміщенні повинна становити не менше 6,0м2, а об’єм не менше 20,0 м3;

— воно повинно мати природне та штучне освітлення відповідно до ДБН В.2.5-28:2018;

— в ньому мають бути шафи для зберігання документів, магнітних дисків, полиці, стелажі, тумби тощо, з урахуванням вимог до площі приміщення;

— щоденно проводити вологе прибирання;

— поруч з приміщенням для роботи з ВДТ мають бути обладнані;

— побутова кімната для відпочинку під час роботи; кімната психологічного розвантаження.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним ВДТ, має здійснюватися системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ [12].

Вимоги до освітлення приміщень та робочих місць під час роботи з ВДТ:

— освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об’єктом розрізнення – найменшим розміром об’єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об’єкта і фону;

— необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

— на робочій поверхні повинні бути відсутні різкі тіні; в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

— величина освітленості повинна бути постійною під час роботи;

— слід обирати оптимальну спрямованість світлового потоку і необхідний склад світла. Застосування світильників без розсіювачів та екрануючих гратів заборонено.

Гігієнічні норми до організації і обладнання робочих місць з ВДТ. При розташуванні елементів робочого місця користувача ВДТ слід враховувати:

— робочу позу користувача;

— простір для розміщення користувача;

— можливість огляду елементів робочого місця;

— можливість ведення захистів;

— розміщення документації і матеріалів, які використовуються користувачем.

4.2 Вимоги до робочого місця

Робоче місце — це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників.

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпусу, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла [13].

Відстань від очей користувача до екрану дисплея має становити 500- 700 мм. Кут зору 10-20°, але не більше 40°; кут між верхнім краєм дисплея і рівнем очей користувача має становити не менше 10°. Кращим є розташування екрану перпендикулярно до лінії зору користувача.

У полі зору користувача ПЕОМ має бути забезпечений відповідний розподіл яскравості. Відношення яскравості екрана до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1 (СНіП 11-4-79). У зв'язку з цим дисплей ПЕОМ повинен відповідати наступним вимогам:

— яскравість свічення екрану не менше 100 кд/м;

— мінімальний розмір світної точки для дисплея не більше 0,6 мм ;

— контрастність зображення знаку – не менше 0,8;

— низькочастотне тремтіння зображення в діапазоні 0,05-1,0 Гц повинно знаходитися в межах 0,1 мм;

— екран повинен мати покриття анти відблиску; відео монітор повинен бути обладнаний поворотним майданчиком, що дозволяє переміщати відео термінал в горизонтальній і вертикальній площинках в межах 130-220 мм і змінювати кут нахилу на 10-15 мм.

Шум несприятливий для людини, особливо при тривалому впливі. В оператора це виражається в зниженні працездатності (наприклад, швидкість обробки тексту зменшується на 10-15%), у прискоренні розвитку зорового стомлення, зміну відчуття кольору, підвищенні витрати енергії (на 17%). Тривалий та інтенсивний шум значно знижує продуктивність праці і призводить до зростання кількості помилок у роботі. У відділі головного економіста шум може створюватися телефонними дзвінками та розмовами, системними блоками та клавіатурою ПЕОМ. Так само джерелами шуму можуть бути системи кондиціювання та вентилювання повітря, існують і зовнішні джерела шуму (наприклад, працюють агрегати на вулиці).

Допустимі рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам «Санітарних норм допустимих рівнів шуму на робочих місцях» № 3223-85. Згідно з цими нормами в приміщенні, де працює користувач ПЕОМ для забезпечення оптимальної робочої середовища рівень шуму не повинен перевищувати 60 дБ. Основними заходами боротьби з шумом згідно з ДСТ 12.1.029-80 ССБТ є ліквідація або ослаблення джерела шуму шляхом застосування звукопоглинаючих матеріалів у конструкціях

механізмів, використання коштів звукопоглинання і раціональна планування виробничого приміщення.

Випромінювання ПК можуть бути небезпечними для здоров'я. Низькочастотні поля при тривалому опроміненні сидять біля ПК людей можуть привести до порушень самих різних фізіологічних процесів. Відповідно до ДСТ 27016-86 і ДСТ 27954-88 потужність дози рентгенівського випромінювання в будь-якій точці простору на відстані 5 см від екрану відео монітора при 41 годинному робочому тижні не повинна перевищувати 100 мкР/год (0,03 мкР/с), а інтенсивність ультрафіолетового випромінювання – 10 Вт/м².

Висока температура повітря негативно позначається на функціональному стані людини. Всі основні електронні блоки ПК мають вбудовані вентилятори для забезпечення стабільних температурних режимів їх функціонування, тому при створенні комфортних умов роботи особливу увагу необхідно приділити шляхам відводу повітря (припливно-витяжної вентиляції).

ВИСНОВКИ

За результатами виконання кваліфікаційної роботи було досягнуто поставлену мету, тобто вирішено практичну задачу інженерії програмного забезпечення, пов’язану з розробкою програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес», що характеризується комплексністю та невизначеністю умов.

Для досягнення мети задачу було розділено на три етапи: перший – аналіз предметної галузі вантажоперевезень; другий – проектування програмного забезпечення, третій – реалізація програмного забезпечення та тестування.

Під час роботи над аналізом предметної галузі було досягнуто таких результатів:

– досліджено організаційну структуру логістичного підприємства «МістЕкспрес» та роботу його диспетчерського відділу;

– досліджено програми-аналоги, що використовуються в подібних логістичних підприємствах та знайдені їх недоліки, що мають бути усунені;

– на основі отриманих знань про предметну галузь та програми-аналоги була обґрунтована доцільність розробки та постановлена задача на розробку програмного забезпечення.

Під час проектування програмного забезпечення було досягнуто таких результатів:

– виконано побудову моделі варіантів використання на основі визначених вимог до програмного забезпечення за допомогою діаграми прецедентів;

– виконано специфікацію основних варіантів використання за допомогою діаграм діяльності;

– створено модель даних програмного забезпечення завдяки концептуальному, логічному та фізичному проектуванню;

– створено проект користувацького інтерфейсу;

– обрана архітектура, на основі якої буде будуватися програмне забезпечення. Архітектурні рішення були відображені на діаграмі компонентів та на статичній моделі класів;

– була побудована динамічна модель програмного забезпечення за допомогою діаграми послідовностей для визначення послідовності подій у процесі роботи програми;

– була побудована діаграма розгортання для пояснення, того, як програмне забезпечення буде розгортатися на сервері.

Під час реалізації програмного забезпечення було досягнуто таких результатів:

– були обрані засоби розробки для кодування програмного забезпечення. В їх число увійшли технологія Java EE, AngularJS та СУБД PostgreSQL;

– була виконана розробка програмного забезпечення за допомогою обраних засобів розробки;

– виконано тестування основних варіантів використання двома методиками тестування: класів еквівалентності і тестуванням варіантів використання.

Процес розробки програмного забезпечення передбачає тривалу роботу за комп’ютером, тому у розділі «Охорона праці» були визначені основні шкідливі фактори при роботі за комп’ютером. З метою послаблення впливу шкідливих факторів рекомендується дотримуватися усіх описаних вимог до організації робочого часу, робочого приміщення та робочого місця.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Державна служба статистики України. «Транспорт і зв'язок України – 2018» / за ред. І. Петренко. Житомир : ТОВ Бук-Друк, 2019. 154 с.

2. Карл Вігерс. Розробка вимог до програмного забезпечення / пер.з англ. Microsoft Corporation, 2004. Київ : ТОВ Ірина-Пресс, 2004. 575 с.

3. РемОнлайн: URL https://remonline.ua/program-for-transport-companies/ (дата звернення: 05.06.2022).

4. ITOB. URL: https://itob.ru/products/1c-tms/ (дата звернення: 05.06.2022).

5. USUSoft URL: http://ususoft.com.ua/uk/programma_dlya_transportnoy_ kompanii.php (дата звернення: 05.06.2022).

6. James Rumbaugh, Ivar Jacobson, Grady Booch. The Unified Modeling Language Reference Manual. Menlo Park : Addison-Wesley, 1998. 568 с.

7. MVC Architecture. URL: https://medium.com/linkit-intecs/mvc-architecture-14238e8b93bc (дата звернення 16.06.2022).

8. Салахалдин Джуба, Андрей Волков. Изучаем PostgreSQL 10. Второе издание. Москва : ДМК-пресс, 2019. 400 с.

9. Differences between Java EE and Java SE. URL:https://docs.oracle.com /javaee/6/firstcup/doc/gkhoy.html (дата звернення 16.06.2022).

10. Діаграма компонентів. URL: http://imlearning.ru/netcat_files/file/FSIS/ ФСИС_семинар-9_Диаграмма-компонентов.pdf (дата звернення 16.06.2022).

11. ISO/IEC TR 19759:2005(E). TECHNICAL REPORT. Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK). [Чинний від 2005-09-15]. Вид. офіц. Женева: Case postale 56 CH-1211, 2005. 210 с.

12. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. Дата оновлення: 27.02.2021. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 07.06.2022).

13. Про охорону здоров’я : Закон України від 19.11.1992 № 2801-XII. Дата оновлення: 31.12.2020. URL: https://zakon.rada.gov.ua/laws/show/2801-12#Text (дата звернення: 07.06.2022).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

Вступ

Найменування програми – програмне забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

Коротка характеристика використання:

Програмне забезпечення призначене для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

1 Підстави для розробки

1.1 Підстави для проведення розробки

Підставою для розробки є наказ ректора на затвердження тем кваліфікаційних робіт бакалаврів.

1.2 Найменування та умовне позначення теми розробки

Найменування розробки – Розробка програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програми є автоматизація роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

2.2 Експлуатаційне призначення

Програма призначена для експлуатації на персональних комп’ютерах та на мобільних пристроях.

3 Вимоги до програми або програмного продукту

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до переліку виконуваних функцій

Програмне забезпечення повинно реалізовувати наступні функції:

– контроль руху вантажівок в режимі реального часу за допомогою інтеграції з GPS провайдерами трекінгових систем;

– ведення обліку клієнтів та їх замовлень;

– призначення незадіяних водіїв та транспорт для виконання замовлення на перевезення;

– створення в системі клієнтів для напрацювання клієнтської бази;

– планування рейсів на майбутнє;

– можливість на основі заповнених диспетчером замовлення та рейсу автоматично генерувати необхідні для дотримання законів в області вантажних перевезень супровідні документи: “Договір про надання транспортних послуг”, “ТТН”, “Маршрутний лист”;

– контроль за виплатами та заборгованостями від клієнтів за надані послуги;

– надання можливості працювати зі збірними вантажами;

– створення рейсів з кількох заявок на перевезення;

– передача інформації про рейси на сторінку водія;

– надання водію можливості змінювати інформацію про рейс під час його виконання;

– автоматична генерація документа “Маршрутний лист” для списання бухгалтерією паливно-мастильних матеріалів;

– отримання звітів з роботи диспетчерського відділу;

– реєстрація диспетчерів та водіїв в програмі і керування їхніми правами доступу.

3.1.2 Вимоги для організації вхідних та вихідних даних

Вхідними даними для роботи диспетчера, адміністратора та водія з програмним забезпеченням є текстова інформація, що вводиться в поля вводу та форми на веб сторінках браузерного клієнта веб-застосунку та записується до бази даних.

Вихідні дані відображаються користувачу на веб-сторінках браузера, також вихідними даними є супровідна документація та звіти що зберігаються в файлах на диску.

3.1.3 Вимоги до часових характеристик

Вимоги до часових характеристик не висуваються.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного функціонування програми

Надійне функціонування програмного забезпечення повинно бути забезпечено виконанням сукупності організаційно-технічних заходів.

3.2.2 Час відновлення після відмови

Час відновлення після відмови, викликаної несправністю технічних засобів, фатальним збоєм операційної системи, не повинен перевищувати часу, необхідного на усунення несправностей технічних засобів і перевстановлення програмних засобів.

3.2.3 Відмови з причини не коректних дій користувача

Дії користувача не впливають на відмови системи.

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови для серверу програмного забезпечення:

– температура повітря в приміщенні: 18-24 °C;

– допустимі відхилення температури: ±2°C;

– відносна вологість повітря: 40-50%.

3.3.2 Вимоги до видів обслуговування

– індикація. Візуально оглядати сервер – якщо який-небудь з апаратних компонентів працює зі збоєм, можна побачити червону лампочку, що горить. Найчастіше виходять з ладу жорсткі диски. Необхідно вчасно помітити проблему, щоб встигнути замінити комплектуючі та уникнути втрати даних;

– пил. Іноді, необхідно чистити сервер від пилу, що накопичився. Для цього рекомендується використовувати балончики зі стисненим повітрям. Особливу увагу варто приділити кулерам;

– читання системних журналів (логів). Найкращий спосіб усунути проблему – не допустити її. За допомогою журналів можна відстежити всі помилки та попередження, які відбуваються в системі та запобігти серйозним наслідкам. Необхідно звернути особливу увагу на системні логи та повідомлення критично важливих програм;

– резервне копіювання. В рамках періодичного обслуговування сервера важливо перевіряти своєчасне створення резервних копій, але що важливіше можливість відновити з них дані.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

Один системний адміністратор.

3.4 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення необхідна наявність у диспетчера та адміністратора персонального комп’ютера, який має такі мінімальні характеристики:

– 8 Гб оперативної пам’яті;

– 150 Гб вільного дискового простору;

– інтегрована в процесор відеокарта;

– дисплей розміром 1536x864 пікселів;

– клавіатура та миша.

Для роботи програмного забезпечення необхідна наявність у водія смартфону.

Для розгортання програмного забезпечення на сервері, який здатен опрацьовувати 50 активних користувачів необхідні два комп’ютера . Сервер веб-застосунку повинен мати такі мінімальні характеристики:

– 16-ти поточний процесор;

– тактова частота процесора 3 ГГц;

– 16 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

Сервер бази даних повинен мати такі мінімальні характеристики:

– 4-х поточний процесор;

– тактова частота процесора 2.5 ГГц;

– 8 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

3.5 Вимоги до інформаційної та програмної сумісності

3.5.1 Вимоги до інформаційних структур та методів розв’язку

Для зберігання даних необхідно використовувати базу даних. В якості СУБД використати PostgreSQL.

3.5.2 Вимоги до вихідного коду та мов програмування

Для розробки веб-застосунку використати останню стабільну версію програмного забезпечення Intellij Idea. Для розробки серверної частини використати технологію Java Enterprice Edition, для розробки клієнтської частини веб-фреймворк AngularJS.

3.5.3 Вимоги до програмних засобів, що використовуються програмою

Системні програмні засоби, що використовуються програмним забезпеченням, повинні бути представлені операційною системою Windows або Linux.

3.5.4 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм не висуваються.

3.6 Вимоги до маркування та упаковки

Програма буде доставлена через мережу інтернет без використання фізичних носіїв, тому вимоги до маркування, упаковки, транспортування та зберігання – не висуваються.

3.6.1 Вимоги до маркування

Вимоги до маркування не висуваються.

3.6.2 Вимоги до упаковки

Вимоги до упаковки не висуваються.

3.6.3 Умови пакування

Вимоги до умов пакування не висуваються.

3.6.4 Порядок пакування

Вимоги до порядку пакування не висуваються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

3.7.1 Умови транспортування та зберігання

Вимоги щодо умов транспортування та зберігання не висуваються.

3.7.2 Спеціальні вимоги

Спеціальні вимоги не висуваються.

4 Вимоги до програмної документації

Склад програмної документації має включати в себе:

1) Технічне завдання.
2) Програму та методику випробувань.
3) Інструкцію користувача.
4) Опис програмного забезпечення.

5 Техніко-економічні показники

Орієнтовна економічна ефективність не розраховується

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програмного забезпечення та строки їх виконання наведені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес»

Стадії розробки	Назва етапу	Початок роботи	Кінець роботи
1	2	3	4
1 Технічне завдання	Аналіз предметної області	01.02.2022	17.02.2022
	Виявлення вимог до програмного забезпечення	18.02.2022	06.03.2022
	Аналіз програм-аналогів	07.03.2022	18.03.2022
	Постановка задачі	19.03.2022	5.04.2022
2 Ескізний проект	Проектування поведінки за допомогою варіантів використання	06.04.2022	10.04.2022
	Специфікація варіантів використання	11.04.2022	12.04.2022
	Розробка концептуальної моделі	13.04.2022	16.04.2022
	Розробка проекту користувальницького інтерфейсу	17.04.2022	20.04.2022
3 Технічний проект	Розробка архітектури	21.04.2022	26.04.2022
	Розробка статичної моделі	27.04.2022	30.04.2022
	Розробка специфікації статичної моделі	01.05.2022	04.05.2022
	Динамічне моделювання поведінки системи	05.05.2022	08.05.2022
	Розробка логічної моделі даних	09.05.2022	10.05.2022
4 Робочий проект	Обґрунтування обраних програмних засобів	11.05.2022	12.05.2022

Продовження таблиці А.1

1	2	3	4
	Моделювання фізичної моделі даних	13.05.2022	14.05.2022
	Моделювання компонентів системи	14.05.2022	16.05.2022
	Моделювання розгортання системи	17.05.2022	18.05.2022
	Реалізація основних класів	19.05.2022	20.05.2022
	Реалізація користувацького інтерфейсу	21.05.2022	23.05.2022
	Тестування	23.05.2022	24.05.2022
	Розробка програми та методики випробувань	25.05.2022	09.06.2022

7 Порядок контролю та прийомки

Прийом програми здійснюється згідно з вимогами програми та методики випробувань, представником замовника та представником виконавця. Результати випробувань заносяться до «Акту прийому програми». У разі невідповідності яким-небудь вимогам технічного завдання, це відображається в «Акті прийому програми» і вимагає доопрацювання розробником програми протягом одного тижня. Після виконання доопрацювання, проводиться повторний прийом програми. Представник замовника повідомляється про повторний прийом не пізніше чотирьох днів до початку випробувань.

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБОВУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

1 Об’єкт випробувань

Об’єктом дослідження є програмне забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

2 Мета випробувань

Мета випробувань – перевірка працездатності програмного забезпечення, перевірка відповідності фактичних характеристик програмного забезпечення тим вимогам, які було викладено у технічному завданні.

3 Вимоги до програмного забезпечення

Під час виконання випробувань перевіряються функціональні характеристики (вимоги), що були викладені в технічному завданні.

4 Вимоги до програмної документації

4.1 Склад програмної документації, що пропонується на випробування Програмна документація включає в себе наступні документи:

1) технічне завдання;

2) програму та методику випробувань;

3) інструкцію користувача;

4) опис програмного забезпечення.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувались під час випробувань:

– Lenovo Legion 5i 15 (AMD Ryzen 5 4600H with Radeon Graphics, 6 ядер, 12 потоків, RAM 16GB, ROM 250TB);

– Samsung Galaxy 22A (RAM 8GB).

5.2 Програмні засоби, що використовувались під час випробувань

Операційна система Windows 10 x64.

Операційна система Android.

5.3 Порядок проведення випробувань

Порядок проведення випробувань – перевірка відповідності функціональних характеристик програмного застосунку, перевірка вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

За відсутності конкретних вимог до оформлення та подання програмної документації, розроблені програмні документи перевіряються представником замовника візуально. Уповноважена особа перевіряє відповідність розроблених програмних документів з тими, що були вказані у пункті «Склад програмної документації, що пропонується на випробування».

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» технічного завдання.

Перевірку можна вважати успішно завершеною у випадку відповідності виконуваних функції технічному завданню.

Процес проведення випробувань мав на меті випробувати функціональність програмного забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес». В таблиці Б.1 наведено перелік виконаних випробувань:

Таблиця Б.1 – Методика випробувань

№	Дія	Очікуваний результат
1	Перевірка можливості авторизації користувача	Після авторизації користувач опиняється на головній сторінці веб-застосунку
2	Перевірка можливості контролю руху вантажівок в режимі реального часу	Програма надає інтерфейс для контролю руху вантажівок в режимі реального часу
3	Перевірка можливості створити замовлення на перевезення	Програма зберігає новостворене замовлення на перевезення
4	Перевірка можливості генерації «ТТН» та «Договору на перевезення»	Програма генерує супровідну документацію та зберігає її на сервері
5	Перевірка можливості створити рейс	Програма зберігає новостворений рейси
6	Перевірка інформативності дошки зайнятості	Програма правильно відображає інформацію на дошці зайнятості
7	Перевірка можливості створити організацію в системі	Програма зберігає новостворену організацію в системі
8	Перевірка можливості створити нову вантажівку в системі	Програма додає нову вантажівку в систему
9	Перевірка можливості реєстрації нового водія	Програма реєструє нового водія в системі
10	Перевірка можливості редагувати поточний рейс водію	Програма дозволяє водію редагувати поточний рейс

ДОДАТОК В – ІНСТРУКЦІЯ ДИСПЕТЧЕРУ ДЛЯ ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

Інструкція користувача призначена для ознайомлення диспетчера логістичного підприємства «МістЕкспрес» з основним функціоналом програмного забезпечення. Інструкція містить основні веб-сторінки веб-застосунку та пояснення до їх функціоналу.

Сторінка авторизації диспетчера.

На рисунку В.1 зображена сторінка авторизації диспетчера. Для авторизації необхідно ввести логін та пароль. Логін та пароль попередньо надаються адміністратором системи.

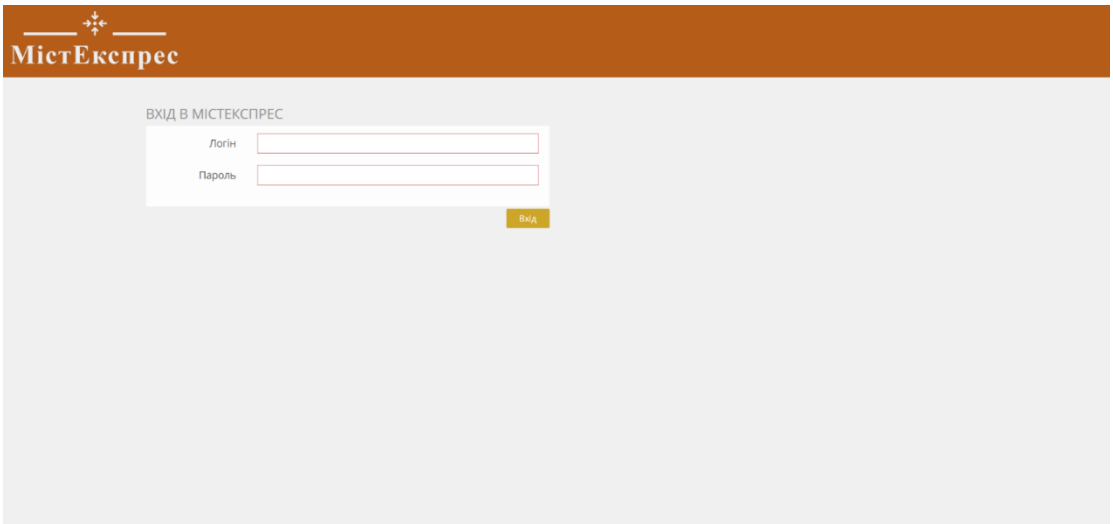


Рисунок В.1 – Сторінка авторизації

Сторінка моніторингу транспорту.

На рисунку В.2 зображена сторінка моніторингу транспорту. Це перша сторінка на яку ви потрапите після авторизації в системі. На цій сторінці ви побачите місцезнаходження транспорту підприємства на даний момент часу.

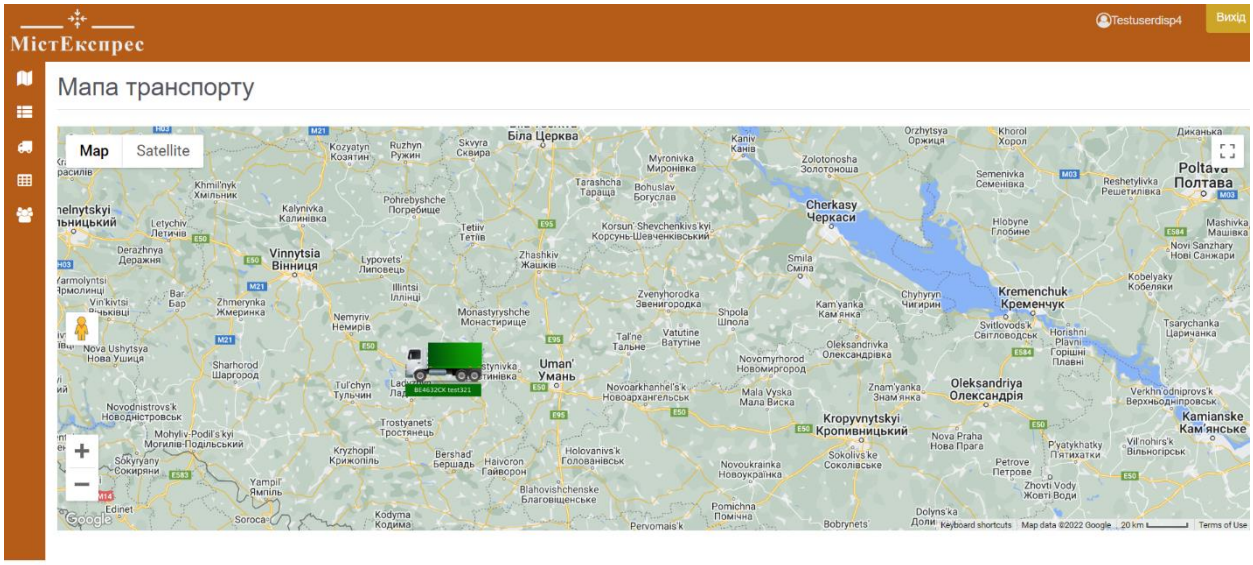


Рисунок В.2 – Сторінка моніторингу транспорту

У правому верхньому куті ви бачите кнопку для виходу з системи та власне ім’я авторизованого диспетчера. Зліва ви побачите меню навігації в системі. Роз’яснення до меню навігації наведено на рисунку В.3.

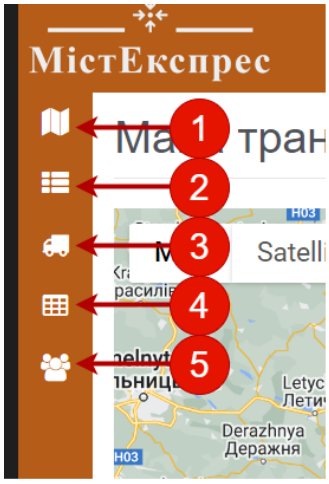


Рисунок В.3 – Меню навігації

Роз’яснення до меню навігації. Перехід на сторінку:

1. «Мапа транспорту»
2. «Дошка замовлень»
3. «Дошка рейсів»
4. «Дошка зайнятості»

5. «Дошка організацій»

Дошка замовлень.

Дошка замовлень відображає список усіх замовлень, що були внесені диспетчером до системи. Дошка замовлень дозволяє відслідковувати процес виконання замовлень на перевезення завдяки зручній фільтрації замовлень за ключовими параметрами. На рисунку В.4 зображена сторінка «Дошка замовлень».

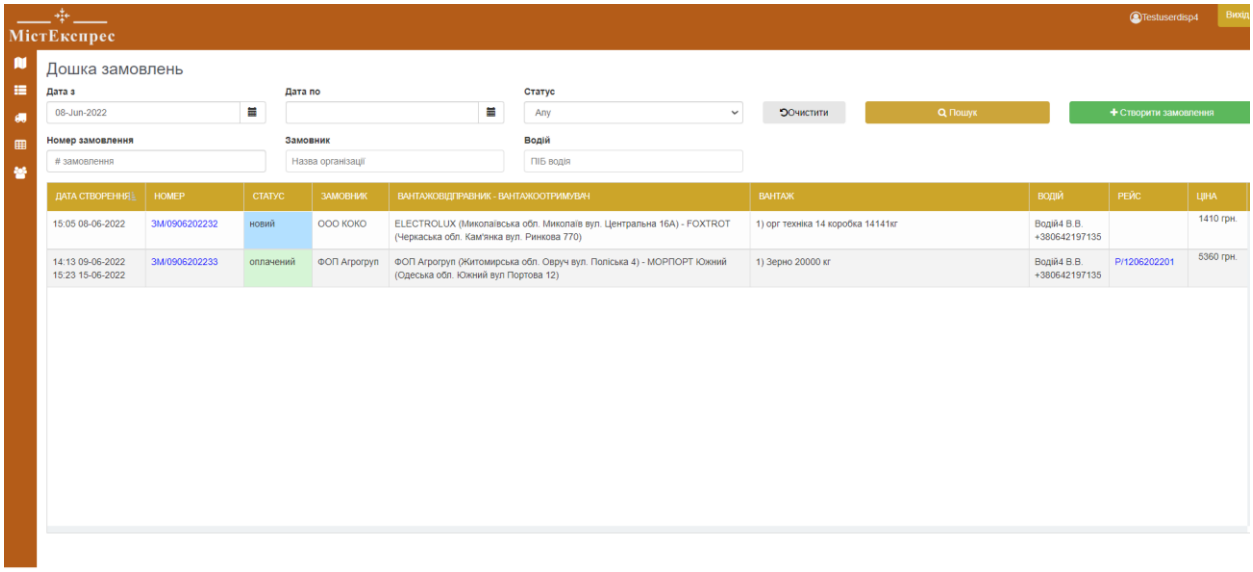


Рисунок В.4 – Сторінка дошки з замовленнями

На сторінці ви бачите фільтр за яким відбувається пошук замовлень на перевезення. Якщо фільтр порожній, то показуються усі замовлення, які були створені за останні три місяці. Якщо буде заповнено поле «Номер замовлення», то пошук буде відбуватися тільки за номером замовлення, інші поля будуть ігноруватися. В таблиці в стовпцях «Номер» і «Рейс» якщо ви клікнете на синій текст відповідно номеру, або рейсу, ви перейдете на сторінку редагування замовлення, або рейсу. Справа зверху ви бачите кнопку «Створити замовлення», натиснувши на неї ви перейдете на сторінку створення нового замовлення.

Створення/редагування замовлення.

На рисунку В.5 зображена сторінка створення/редагування замовлення.

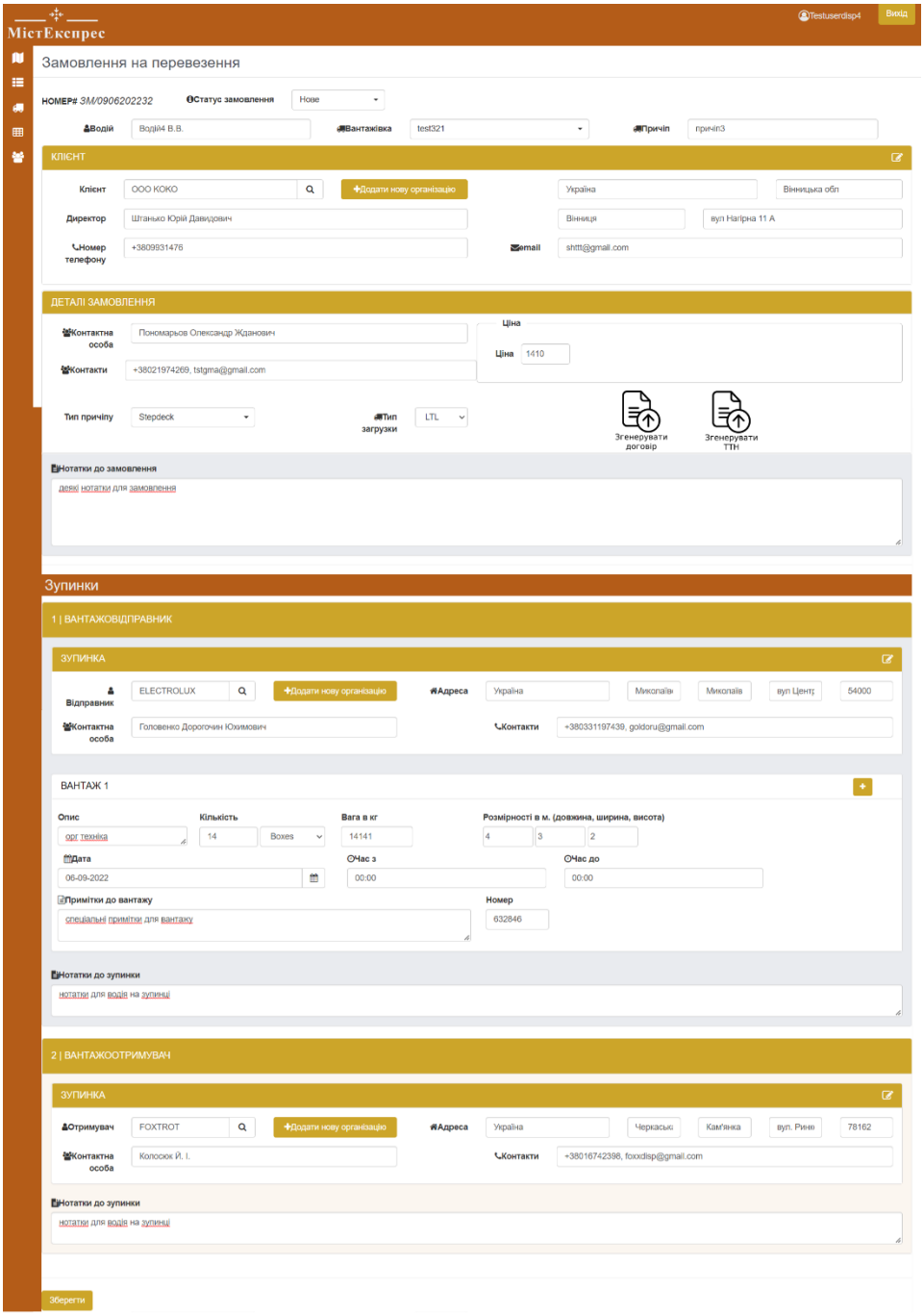


Рисунок В.5 – Сторінка створення/редагування замовлення

Сторінка поділена на 6 розділів: виконавець, клієнт, деталі замовлення, вантажовідправник, вантаж, вантажоотримувач. Кожен розділ відповідає за окремий аспект вантажоперевезення. Примітки до сторінки створення/редагування замовлення на перевезення вказані на рисунку В.6.

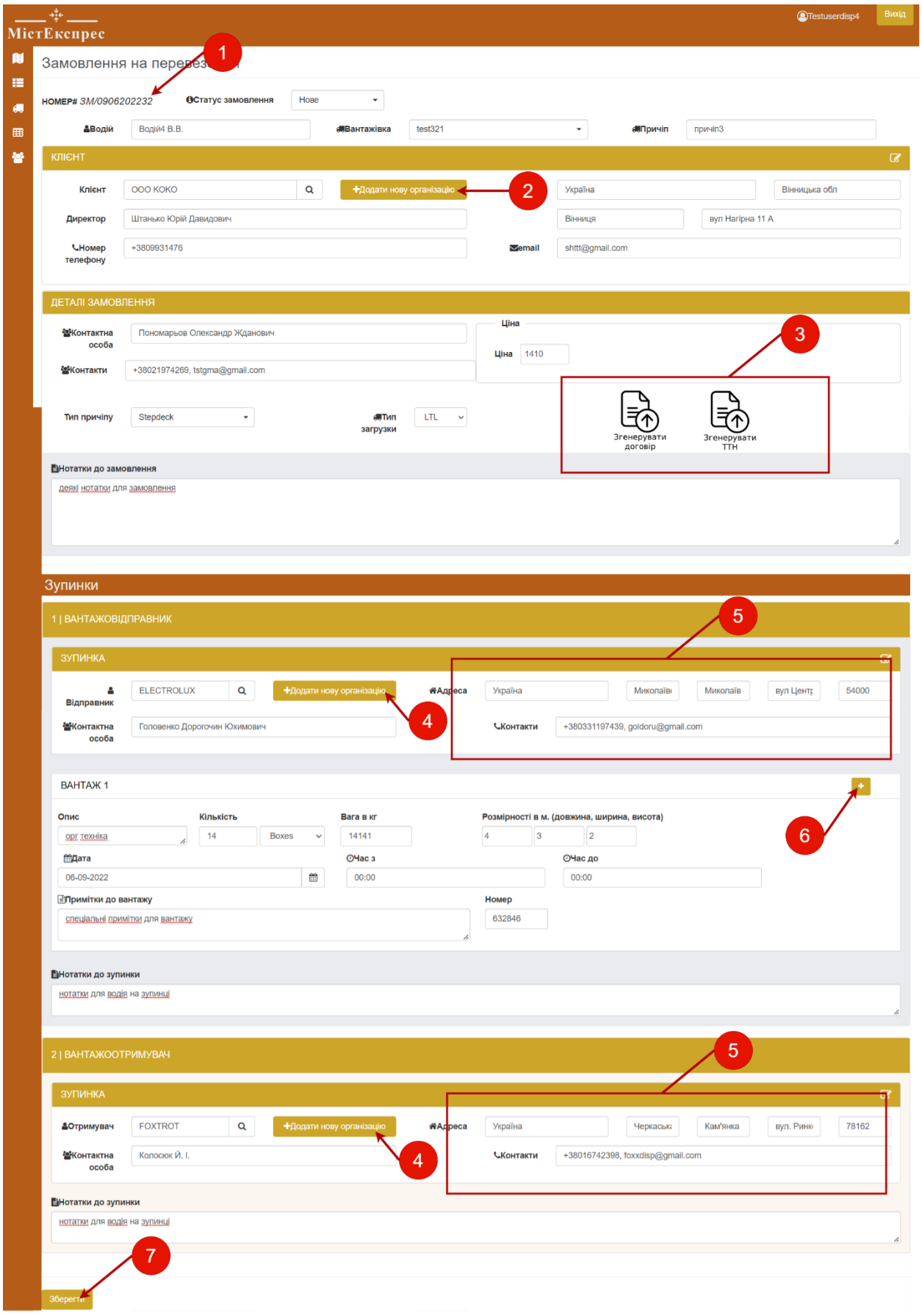


Рисунок В.6 – Примітки до сторінки створення/редагування замовлення

Примітки до сторінки створення/редагування замовлення:

1. Після збереження замовлення, замовленню автоматично присвоюється згенерований номер замовлення.

2. Якщо клієнт не знайдений в системі, тоді ви можете натиснути кнопку «Додати нову організацію», після цього відкриється сторінка створення організації де ви зможете додати нову організацію і повернутися назад. Якщо ж клієнт вже існує в системі то вам потрібно лише знайти його за назвою організації, усі інші поля заповняться автоматично.

3. Функціонал з генерації та подальшого перегляду супровідної документації стане доступним після збереження замовлення на перевезення та присвоєння йому статусу «Готовий до рейсу».

4. Вантажовідправник та вантажоотримувач теж є організаціями, тому для них застосована таж сама логіка, що й до клієнта з пункту 2.

5. Для того, щоб вказати адресу завантаження або доставки ви повинні почати друкувати її у відведені поля, якщо місце для організації вантажовідправника або вантажоотримувача існує, то воно з’явиться в полі в якості списку, якщо місця в системі немає ви можете клікнути на кнопку справа зверху виділеної області 5 та додати місце до організації, а потім вже додати його до замовлення.

6. Натиснувши цю кнопку відкриється ще один розділ для додаткового вантажу.

7. Після натискання за кнопку збереження програма перевірить замовлення на коректність. Буде перевірятися: час завантаження та доставки, відповідність причіпу для перевезення вантажу, можливість водія виконати замовлення на вказаному транспорті у вказаний проміжок часу.

Якщо помилок не буде знайдено, замовлення на перевезення збережеться зі статусом «Готовий до рейсу». Якщо помилка буде знайдена, ви отримаєте попередження про помилку, але все одно зможете зберегти замовлення без зміни статусу, що унеможливіть подальшу роботу з замовленням.

ДОДАТОК Г – ТЕКСТ ПРОГРАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

```
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app>

    <display-name>Archetype Created Web Application</display-name>

    <context-param>
        <param-name>APP_LINK</param-name>
        <param-value>http://localhost:8080/app</param-value>
    </context-param>

    <resource-ref>
        <res-ref-name>jdbc/app</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>

    <listener>
        <listener-class>app.AppStartup</listener-class>
    </listener>

    <error-page>
        <error-code>401</error-code>
        <location>/html/error/401.html</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/html/error/404.html</location>
    </error-page>

    <servlet>
        <servlet-name>LoginLogoutServlet</servlet-name>
        <servlet-class>app.servlet.LoginLogoutServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginLogoutServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>ACLUserServlet</servlet-name>
        <servlet-class>app.servlet.ACLUserServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ACLUserServlet</servlet-name>
        <url-pattern>/user</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>OrganizationServlet</servlet-name>
```

```
        <servlet-class>app.servlet.OrganizationServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>OrganizationServlet</servlet-name>
        <url-pattern>/organization</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>OrderServlet</servlet-name>
        <servlet-class>app.servlet.OrderServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>OrderServlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>TripServlet</servlet-name>
        <servlet-class>app.servlet.TripServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TripServlet</servlet-name>
        <url-pattern>/trip</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DriverServlet</servlet-name>
        <servlet-class>app.servlet.DriverServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DriverServlet</servlet-name>
        <url-pattern>/driver</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>Truck</servlet-name>
        <servlet-class>app.servlet.Truck</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Truck</servlet-name>
        <url-pattern>/truck</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>TrailerServlet</servlet-name>
        <servlet-class>app.servlet.TrailerServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TrailerServlet</servlet-name>
        <url-pattern>/trailer</url-pattern>
    </servlet-mapping>

</web-app>


package app;
import javax.naming.NamingException;
import java.io.IOException;
import java.util.logging.Logger;


public class AppStartup implements ServletContextListener {
    private static final Logger log = Logger.getLogger(AppStartup.class);
    @Override
```

```
    public void contextInitialized(ServletContextEvent sce) {
        try {
            ServletConstants.APP_LINK =
sce.getServletContext().getInitParameter("APP_LINK");
            AppConstants.initializeProperties();
            PGDBPool.initializeConnectionPool();
            ArgonInitialize.getInstance();
        } catch (IOException | NamingException e) {
            log.error("Cannot initialize application", e);
        }
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {}
}
```

```
package app;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.logging.Logger;

public class PGDBPool {
    private static final Logger log = Logger.getLogger(PGDBPool.class);
    private static DataSource pool;
    private static String path = "java:comp/env/jdbc/dispapp";
    private static String username = "vladdr";
    private static String password = "511770";


    public static void initializeConnectionPool() throws NamingException {
        InitialContext initContext = new InitialContext();
        pool = (DataSource) initContext.lookup(path, username, password);
    }

    public static Connection getConnectionFromPool() throws SQLException {
        try {
            return pool.getConnection();
        } catch (SQLException e) {
            log.error("PGDBPool Cannot get connection from pool", e);
            throw e;
        }
    }

    public static void releaseConnection(Connection connection) {
        try {
            if (connection != null) connection.close();
        } catch (SQLException e) {
            log.error("PGDBPool Cannot release connection", e);
        }
    }
}
```

```
package app;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.util.Collection;
import java.util.logging.Logger;

public class HttpUtil {
```

```
    private static final Logger log = Logger.getLogger(HttpUtil.class);

    public HttpUtil(HttpServletRequest request, HttpServletResponse response)
{
        this.request = request;
        this.response = response;
    }

    private HttpServletRequest request;
    private HttpServletResponse response;

    public Long getSessionUserId() {
        return (Long) request.getSession(false)
                .getAttribute(ServletConstants.ATTRIBUTE_NAME_USER_ID);
    }
    public String getParameter(String parameterName) {
        return request.getParameter(parameterName);
    }

    public <OUT>void sendResponse(int statusCode, Collection<OUT>
dtoCollection) {
        response.setCharacterEncoding("UTF-8");
        ObjectMapper mapper = new ObjectMapper();
        try {
            PrintWriter writer = response.getWriter();
            response.setStatus(statusCode);
            mapper.writeValue(writer, dtoCollection);
        } catch (Exception e) {
            log.error("Error in sendDTOCollection()", e);
        }
    }
    public <OUT>void sendResponse(int statusCode, String str) {
        response.setCharacterEncoding("UTF-8");
        ObjectMapper mapper = new ObjectMapper();
        try {
            PrintWriter writer = response.getWriter();
            response.setStatus(statusCode);
            mapper.writeValue(writer, str);
        } catch (Exception e) {
            log.error("Error in sendDTOCollection()", e);
        }
    }
    public <OUT>void sendResponse(int statusCode, Object obj) {
        response.setCharacterEncoding("UTF-8");
        ObjectMapper mapper = new ObjectMapper();
        try {
            PrintWriter writer = response.getWriter();
            response.setStatus(statusCode);
            mapper.writeValue(writer, obj);
        } catch (Exception e) {
            log.error("Error in sendDTOCollection()", e);
        }
    }
    public <IN>IN deserializeData(Class<IN> dtoClass) {
        ObjectMapper mapper = new ObjectMapper();
        try {
            String json = parseJSONString();
            return (json != null) ? mapper.readValue(json, dtoClass) : null;
        } catch (JsonProcessingException e) {
            log.error("Error in deserializeDTO()", e);
            return null;
        }
    }
```

```
    private String parseJSONString() {
        try (BufferedReader reader = this.request.getReader()) {
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
            return sb.toString();
        } catch (Exception e) {
            log.error("Exception in parseJSONString: " + e.getMessage(), e);
            return null;
        }
    }

    public HttpServletRequest getRequest() {
        return request;
    }

    public HttpServletResponse getResponse() {
        return response;
    }
}


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="libs/bootstrap-4.5.2-
dist/css/bootstrap.min.css">
    <!-- Popper JS -->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.j
s"></script>
    <!-- jQuery library -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scri
pt>
    <!-- Latest compiled JavaScript -->
    <script src="libs/bootstrap-4.5.2-dist/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/main-style.css">
    <link rel="stylesheet" href="css/article.css">

    <script src="libs/angular-1.8.0/angular-animate.js"></script>
    <script src="js/appModule.module.js"></script>
    <script src="js/appConstants.component.js"></script>
    <script src="js/index.component.js"></script>

</head>

<!-- Modal log -->
<div id="logModal" class="modal fade" role="dialog">
    <div class="modal-dialog">

        <!-- Modal content-->
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">Вхід</h4>
                <button type="button" class="close" data-
dismiss="modal">&times;</button>
            </div>
```

```
            <form class="form" name="loginForm">
                <div class="modal-body">
                    <div class="form-group">
                        <input name="name" ng-model="vm.loginUser.login"
type="email" class="form-control"  placeholder="email" required>
                    </div>
                    <div class="form-group">
                        <input name="password" ng-
model="vm.loginUser.password" type="password" class="form-control"
placeholder="пароль" required>
                    </div>
                </div>

                <div class="modal-footer">
                    <button type="submit" ng-click="vm.loginUser(loginForm)"
name="submit" class="btn btn-submit" value="submit">Увійти</button>
                </div>

            </form>
        </div>
    </div>
</div>

</html>


(function () {
    'use strict';

    angular
        .module('appModule')
        .controller('indexController', Controller);

    Controller.$inject = ['$http', '$cookies', '$uibModal', 'APP_LINK'];

    function Controller($http, $cookies, $uibModal, APP_LINK) {
        const vm = this;

        vm.isLogin = false;
        vm.currentUser = {};

        vm.loginUser = {
            login: null,
            password: null
        };

        vm.loginUser = loginUser;
        vm.invalidateSession = invalidateSession;

        activate();

        function activate() {
            getLoginUser();
        }

        function getLoginUser() {
            $http.post(APP_LINK + 'app/user?action=getUserById')
                .then((response => {
                    console.log('RESPONSE:', response);
                    vm.isLogin = true;
                    vm.currentUser = response.data;
                }), (() => {}));
        }
```

```
        function loginUser(form) {
            if (!form.$valid) return;
            $http.post(APP_LINK +
'login?action=login&login='+vm.loginUser.login+'&password='+vm.loginUser.pass
word)
                .then((response => {
                    console.log('RESPONSE:', response);
                    window.location.href = "html/incomeBook.component.html";
                }), (
                    (response) => {
                        if (response.status === 401) {
                            alert('Невірний логін або пароль');
                        } else {
                            alert('Щось пішло не так!');
                        }
                    }));
        }

        function invalidateSession() {
            $http.post(APP_LINK + 'login?action=logout')
                .then(
                    () => {
                        window.location.href = "index.jsp"; //This line
redirects (redirect in Java does not work? because this method $http.get()
асинхроннij).
                    },
                    () => {
                        alert('Виникла помилка. Повідомте будь ласка
розробників.');
                    }
                );
        }

    }
})();


package app.servlet;
import app.HttpUtil;
import app.delegate.LoginLogoutDelegate;
import app.vo.ACLUser;

import java.io.UnsupportedEncodingException;
import java.util.logging.Logger;

public class LoginLogoutServlet extends HttpServlet {
    private static final Logger log =
Logger.getLogger(LoginLogoutServlet.class);
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
{
        HttpUtil su = new HttpUtil(req, resp);
        try {
            su.getRequest().setCharacterEncoding("UTF-8");
        } catch (UnsupportedEncodingException e) {
            log.error(e);
        }
        String action = req.getParameter(ServletConstants.ACTION);
        switch (action) {
            case ServletConstants.LOGIN : handleLoginRequest(su); break;
            case ServletConstants.LOGOUT : handleLogoutRequest(su); break;
            default: su.sendResponse(500, "Server error");
        }
    }
```

```
    private void handleLoginRequest(HttpUtil su) {
        try {
            String login = su.getRequest().getParameter("login");
            String password = su.getRequest().getParameter("password");

            LoginLogoutDelegate loginLogoutDelegate = new
LoginLogoutDelegate();
            ACLUser aclUser = loginLogoutDelegate.loginUser(login, password);

            if (aclUser == null) {
                log.info("Access denied for: login name = " + login + " ,
password = " + password);
                su.sendResponse(401, "Неправильний логін або пароль");
                return;
            }

            HttpSession session = su.getRequest().getSession(true);
            session.setAttribute(ServletConstants.ATTRIBUTE_NAME_USER_ID,
aclUser.getId());

            Cookie languageCookie = new
Cookie(ServletConstants.COOKIE_NAME_LANGUAGE, language);
            languageCookie.setMaxAge(-1); // до кінця сесії
            su.getResponse().addCookie(languageCookie);

            log.info("User logged in: login name = " + login + " , password =
" + password);
            su.getResponse().sendRedirect(ServletConstants.APP_LINK +
"app/transport-map");
        } catch (Exception e) {
            log.error("SERVLET LOGIN Cannot login session. actionLogin()",
e);
            su.sendError(e.getMessage());
        }
    }

    private void handleLogoutRequest(HttpUtil su) {
        try {
            LoginLogoutDelegate userDelegate = new LoginLogoutDelegate();
            userDelegate.logoutUser(su);
        } catch (Exception e) {
            log.error("SERVLET LOGOUT Cannot logout session. actionLogout()",
e);
            su.sendError(e.getMessage());
        }
    }

}


package app.servlet;
import app.HttpUtil;
import app.delegate.OrderDelegate;
import app.vo.Order;

import java.io.UnsupportedEncodingException;
import java.util.logging.Logger;

public class OrderServlet extends HttpServlet{
    private static final Logger log =
Logger.getLogger(LoginLogoutServlet.class);
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
```

```
{
        HttpUtil su = new HttpUtil(req, resp);
        try {
            su.getRequest().setCharacterEncoding("UTF-8");
        } catch (UnsupportedEncodingException e) {
            log.error(e);
        }
        String action = req.getParameter(ServletConstants.ACTION);
        switch (action) {
            case ServletConstants.CREATE_ORDER : handleCreateOrder(su);
break;
            case ServletConstants.EDIT_ORDER : handleEditOrder(su); break;
            default: su.sendResponse(500, "Server error");
        }
    }

    private void handleCreateOrder(HttpUtil su) {
        try {
            Order order = su.deserializeData(Order.class);
            OrderDelegate orderDelegate = new OrderDelegate();
            Long orderId = orderDelegate.createOrder(order);
            su.sendResponse(200, Long.toString(orderId));
        } catch (Exception e) {
            log.error("Cannot create order", e);
            su.sendError(e.getMessage());
        }
    }

    private void handleEditOrder(HttpUtil su) {
        try {
            Order order = su.deserializeData(Order.class);
            OrderDelegate orderDelegate = new OrderDelegate();
            Order orderEd = orderDelegate.updateOrder(order);
            su.sendResponse(200, orderEd);
        } catch (Exception e) {
            log.error("Cannot update order", e);
            su.sendError(e.getMessage());
        }
    }

}


package app.delegate;
import app.PGDBPool;
import java.sql.Connection;
import java.sql.SQLException;

public class Delegate {
    final Connection connection;
    private final boolean classManageConnection;
    private boolean finalizeControl = true;
    private boolean inTransaction;

    public Delegate() throws SQLException {
        this.connection = PGDBPool.getConnectionFromPool();
        classManageConnection = true;
    }
    public Delegate(Connection connection) {
        this.connection = connection;
        classManageConnection = false;
    }

    public boolean setFinalizeControl() {
```

```
        if (finalizeControl) {
            finalizeControl = false;
            return true;
        } else {
            return false;
        }
    }

    public void startTransaction() throws SQLException {
        if (!inTransaction && classManageConnection) {
            connection.setAutoCommit(false);
            inTransaction = true;
        }
    }

    public void rollbackTransaction() throws SQLException {
        if (inTransaction && classManageConnection) {
            connection.rollback();
        }
    }

    public void commitTransaction() throws SQLException {
        if (classManageConnection) {
            connection.commit();
            inTransaction = false;
        }
    }

    public void releaseConnection(boolean release) {
        if (classManageConnection) {
            if (release) {
                PGDBPool.releaseConnection(connection);
            }
        }
    }

}


package app.delegate;
import java.sql.Connection;
import app.dao.PGACLUserDAO;
import app.vo.ACLUser;
import java.sql.SQLException;
import java.util.logging.Logger;

public class ACLUserDelegate extends Delegate {
    private static final Logger log =
Logger.getLogger(ACLUserDelegate.class);

    public ACLUserDelegate(Connection connection) {
        super(connection);
    }
    public ACLUserDelegate() throws SQLException {
        super();
    }

    public Long saveACLUser(ACLUser userVO) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            PGACLUserDAO userDAO = new PGACLUserDAO();
            return userDAO.addACLUser(connection, userVO);
        } catch (Exception e) {
            log.error("DELEGATE Cannot add new acl_user. addUser()", e);
```

```
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

    public Long updateACLUser(Long userId, ACLUser userVO) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            PGACLUserDAO userDAO = new PGACLUserDAO();
            return userDAO.updateACLUser(connection, userVO);
        } catch (Exception e) {
            log.error("DELEGATE Cannot update acl_user. updateUser()", e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

    public ACLUser getACLUserById(Long userId) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            PGACLUserDAO userDAO = new PGACLUserDAO();
            return userDAO.getACLUserById(connection, userId);
        } catch (Exception e) {
            log.error("DELEGATE Cannot get acl_user by id. getUserById()",
e);
            throw e;
        } finally {
          releaseConnection(rc);
        }
    }

    public ACLUser getACLUserByLogin(String login) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            PGACLUserDAO userDAO = new PGACLUserDAO();
            return userDAO.getACLUserByLogin(connection, login);
        } catch (Exception e) {
            log.error("DELEGATE Cannot get acl_user by id. getUserById()",
e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

}


package app.delegate;
import app.HttpUtil;
import java.sql.Connection;
import app.vo.ACLUser;

import java.sql.SQLException;

public class LoginLogoutDelegate extends Delegate {

    public LoginLogoutDelegate() throws SQLException {
        super();
    }
    public LoginLogoutDelegate(Connection connection) {
        super(connection);
```

```
    }

    public ACLUser loginUser(String loginName, String password) throws
Exception {
        boolean rc = setFinalizeControl();
        try {
            ACLUserDelegate aclUserDelegate = new
ACLUserDelegate(connection);
            ACLUser aclUser = aclUserDelegate.getACLUserByLogin(loginName);

            String passwordDBHash = aclUser.getPassword();
            boolean isPasswordMatch = checkPassword(passwordDBHash,
password);

            if (!isPasswordMatch) {
                return null;
            } else {
                return aclUser;
            }
        } catch (Exception e) {
            log.error("DELEGATE Cannot login acl_user with " +
                    "loginName = " + loginName + " and password = " +
password + ". updateUser()", e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

    public void logoutUser(HttpUtil su) {
        su.getRequest().getSession().invalidate();
    }

    private boolean checkPassword(String passwordDBHash, String
inputPassword) {
        return ArgonInitialize.getInstance()
                .getArgon2().verify(passwordDBHash, inputPassword);
    }
}


package app.delegate;
import java.sql.Connection;
import app.dao.PGOrderDAO;
import app.vo.Order;
import javafx.scene.paint.Stop;
import java.sql.SQLException;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

public class OrderDelegate extends Delegate {
    public OrderDelegate() throws SQLException {
        super();
    }
    public OrderDelegate(Connection connection) {
        super(connection);
    }

    public Long createOrder(Order order) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            startTransaction();
            CargoDelegate cargoDelegate = new CargoDelegate(connection);
```

```
            StopDelegate stopDelegate = new StopDelegate(connection);
            Long orderId = saveOrder(connection, order);
            List<Cargo> cargoList = order.getCargoList();
            List<Stop> stopList = order.getStopList();
            cargoList.forEach(c -> {
                c.setOrderId(orderId);
            });
            stopList.forEach(s -> {
                s.setOrderId(orderId);
            });
            cargoDelegate.addCargo(connection, cargoList);
            stopDelegate.addStops(connection, stopList);
            commitTransaction();
            return orderId;
        } catch (Exception e) {
            rollbackTransaction();
            log.error("DELEGATE Cannot create order", e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

    private Long saveOrder(Order order) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            PGOrderDAO pgOrderDAO = new PGOrderDAO();
            return pgOrderDAO.addOrder(connection, order);
        } catch (Exception e) {
            log.error("DELEGATE Cannot save order", e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

    public Order updateOrder(Order order) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            startTransaction();
            CargoDelegate cargoDelegate = new CargoDelegate(connection);
            StopDelegate stopDelegate = new StopDelegate(connection);
            Long orderId = orderDelegate.updateOrder(order);
            List<Cargo> cargoList = order.getCargoList();
            List<Stop> stopList = order.getStopList();
            List<Cargo> cargoListExisted =
cargoDelegate.getCargoListByOrderId(connection, orderId);
            Set<Long> cargoIds = cargoListExisted.stream().map(x = x ->
x.getId()).collect(Collectors.toList());
            List<Stop> stopListExisted =
stopDelegate.getStopListByOrderId(connection, orderId);
            Set<Long> stopIds = stopListExisted.stream().map(x = x ->
x.getId()).collect(Collectors.toList());
            cargoDelegate.deleteCargoByIdSet(connection, cargoIds);
            stopDelegate.deleteStopByIdSet(connection, stopIds);
            cargoDelegate.addCargo(connection, cargoList);
            stopDelegate.addStops(connection, stopList);
            commitTransaction();
            return order;
        } catch (Exception e) {
            rollbackTransaction();
            log.error("DELEGATE Cannot update order", e);
            throw e;
        } finally {
```

```
            releaseConnection(rc);
        }
    }

    public Order getOrderByOrderId(Long orderId) throws Exception {
        boolean rc = setFinalizeControl();
        try {
            startTransaction();
            CargoDelegate cargoDelegate = new CargoDelegate(connection);
            StopDelegate stopDelegate = new StopDelegate(connection);
            Order order = new PGOrderDAO().getOrderByOrderId(connection,
orderId);
            List<Cargo> cargoList =
cargoDelegate.getCargoListByOrderId(connection, orderId);
            List<Stop> stopList =
stopDelegate.getStopListByOrderId(connection, orderId);
            order.setCargoList(cargoList);
            order.setCargoList(stopList);
            commitTransaction();
            return order;
        } catch (Exception e) {
            rollbackTransaction();
            log.error("DELEGATE Cannot getOrderByOrderId", e);
            throw e;
        } finally {
            releaseConnection(rc);
        }
    }

}


package app.dao;
import java.sql.Connection;
import app.vo.ACLUser;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class PGACLUserDAO {
    public Long addACLUser(Connection connection, ACLUser vo) throws
SQLException {
        String sql = "INSERT INTO acl_user (id, login, password, type,
status, register_date) " +
                "VALUES (?, ?, ?, ?, ?, ?) RETURNING id;";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, vo.getId());
            ps.setString(2, vo.getLogin());
            ps.setString(3, vo.getPassword());
            ps.setString(4, vo.getType());
            ps.setString(5, vo.getStatus());
            ps.setDate(6, vo.getRegisterDate());
            ResultSet rs = ps.executeQuery();
            Long userId = null;
            while (rs.next()) {
                userId = rs.getLong("id");
            }
            return userId;
        } catch (SQLException e) {
            log.error("DAO Cannot add new user to DB. addUser()", e);
            throw e;
        }
    }
```

```
    public Long updateACLUser(Connection connection, ACLUser vo) throws
SQLException {
        String sql = "UPDATE acl_user SET login = ?, password = ?, type = ?,
status = ?, register_date = ? " +
                "WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, vo.getLogin());
            ps.setString(2, vo.getPassword());
            ps.setString(3, vo.getType());
            ps.setString(4, vo.getStatus());
            ps.setLong(5, vo.getId());
            return ps.executeUpdate();
        } catch (SQLException e) {
            log.error("DAO Cannot update user to DB. updateUser()", e);
            throw e;
        }
    }

    public ACLUser getACLUserById(Connection connection, Long userId) throws
SQLException {
        String sql = "SELECT * FROM acl_user WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, userId);
            ResultSet rs = ps.executeQuery();
            ACLUser userVO = null;
            if (rs.next()) userVO = new ACLUser(rs);
            return userVO;
        } catch (SQLException e) {
            log.error("DAO Cannot get user by id from DB. getUserById()", e);
            throw e;
        }
    }

    public ACLUser getACLUserByLogin(Connection connection, String login)
throws SQLException {
        String sql = "SELECT * FROM acl_user WHERE login = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, login);
            ResultSet rs = ps.executeQuery();
            ACLUser userVO = null;
            if (rs.next()) userVO = new ACLUser(rs);
            return userVO;
        } catch (SQLException e) {
            log.error("DAO Cannot get user by id from DB. getUserById()", e);
            throw e;
        }
    }

}


package app.dao;
import java.sql.Connection;
import app.vo.Order;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class PGOrderDAO {
    public Long addOrder(Connection connection, Order vo) throws SQLException
{
        String sql = "INSERT INTO order (id, order_number, trip_id,
client_id, dispatcher_id, " +
                "driver_id, truck_id, trailer_id, status, open_date,
```

```
close_date, load_type, price, " +
                "notes, client_id, dispatcher_id, driver_id, truck_id,
trailer_id) " +
                "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?) RETURNING id;";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, vo.getId());
            ps.setString(2, vo.getOrderNumber());
            ps.setLong(3, vo.getTripId());
            ps.setLong(4, vo.getClientId());
            ps.setLong(5, vo.getDispatcherId());
            ps.setLong(6, vo.getDriverId();
            ps.setLong(7, vo.getTruckId();
            ps.setLong(8, vo.getTrailerId();
            ps.setString(9, vo.getStatus();
            ps.setDate(10, vo.getOpenDate();
            ps.setDate(11, vo.getCloseDate();
            ps.setString(12, vo.getLoadType();
            ps.setLong(13, vo.getPrice();
            ps.setString(14, vo.getNotes();
            ps.setLong(15, vo.getClientId();
            ps.setLong(16, vo.getDispatcherId();
            ps.setLong(17, vo.getTruckId();
            ps.setLong(18, vo.getTrailerId();
            ResultSet rs = ps.executeQuery();
            Long userId = null;
            while (rs.next()) {
                userId = rs.getLong("id");
            }
            return userId;
        } catch (SQLException e) {
            log.error("DAO Cannot add new order to DB", e);
            throw e;
        }
    }

    public Long updateOrder(Connection connection, Order vo) throws
SQLException {
        String sql = "UPDATE order SET id=?, order_number=?, trip_id=?,
client_id=?, dispatcher_id=?, " +
                "driver_id=?, truck_id=?, trailer_id=?, status=?,
open_date=?, close_date=?, load_type=?, price=?, " +
                "notes=?, client_id=?, dispatcher_id=?, driver_id=?,
truck_id=?, trailer_id=? " +
                "RETURNING id;";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, vo.getId());
            ps.setString(2, vo.getOrderNumber());
            ps.setLong(3, vo.getTripId());
            ps.setLong(4, vo.getClientId());
            ps.setLong(5, vo.getDispatcherId());
            ps.setLong(6, vo.getDriverId());
            ps.setLong(7, vo.getTruckId());
            ps.setLong(8, vo.getTrailerId());
            ps.setString(9, vo.getStatus());
            ps.setDate(10, vo.getOpenDate());
            ps.setDate(11, vo.getCloseDate());
            ps.setString(12, vo.getLoadType());
            ps.setLong(13, vo.getPrice());
            ps.setString(14, vo.getNotes());
            ps.setLong(15, vo.getClientId());
            ps.setLong(16, vo.getDispatcherId());
            ps.setLong(17, vo.getTruckId());
            ps.setLong(18, vo.getTrailerId());
            ResultSet rs = ps.executeQuery();
```

```
            Long userId = null;
            while (rs.next()) {
                userId = rs.getLong("id");
            }
            return userId;
        } catch (SQLException e) {
            log.error("DAO Cannot add new order to DB", e);
            throw e;
        }
    }

    public Long getOrderByOrderId(Connection connection, Long id) throws
SQLException {
        String sql = "SELECT * FROM order WHERE id=?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setLong(1, id);
            ResultSet rs = ps.executeQuery();
            return new Order(rs);
        } catch (SQLException e) {
            log.error("DAO Cannot add new order to DB", e);
            throw e;
        }
    }

}


package app.vo;

import java.util.Date;

public class ACLUser {
    private Long id;
    private String login;
    private String password;
    private String type;
    private String status;
    private Date registerDate;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public String getStatus() {
```

```
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public Date getRegisterDate() {
        return registerDate;
    }
    public void setRegisterDate(Date registerDate) {
        this.registerDate = registerDate;
    }
}
package app.vo;
import app.vo.*;
import java.util.Date;

public class Order {
    private Long id;
    private String orderNumber;
    private Long tripId;
    private Long clientId;
    private Long dispatcherId;
    private Long driverId;
    private Long truckId;
    private Long trailerId;
    private String status;
    private Date openDate;
    private Date closeDate;
    private String loadType;
    private Integer price;
    private String notes;
    private Organization client;
    private Dispatcher dispatcher;
    private Driver driver;
    private Truck truck;
    private Trailer trailer;
    private List<Stop> stopList;
    private List<Cargo> cargoList;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getOrderNumber() {
        return orderNumber;
    }
    public void setOrderNumber(String orderNumber) {
        this.orderNumber = orderNumber;
    }
    public Long getTripId() {
        return tripId;
    }
    public void setTripId(Long tripId) {
        this.tripId = tripId;
    }
    public Long getClientId() {
        return clientId;
    }
    public void setClientId(Long clientId) {
        this.clientId = clientId;
    }
    public Long getDispatcherId() {
```

```
    return dispatcherId;
}
public void setDispatcherId(Long dispatcherId) {
    this.dispatcherId = dispatcherId;
}
public Long getDriverId() {
    return driverId;
}
public void setDriverId(Long driverId) {
    this.driverId = driverId;
}
public Long getTruckId() {
    return truckId;
}
public void setTruckId(Long truckId) {
    this.truckId = truckId;
}
public Long getTrailerId() {
    return trailerId;
}
public void setTrailerId(Long trailerId) {
    this.trailerId = trailerId;
}
public String getStatus() {
    return status;
}
public void setStatus(String status) {
    this.status = status;
}
public Date getOpenDate() {
    return openDate;
}
public void setOpenDate(Date openDate) {
    this.openDate = openDate;
}
public Date getCloseDate() {
    return closeDate;
}
public void setCloseDate(Date closeDate) {
    this.closeDate = closeDate;
}
public String getLoadType() {
    return loadType;
}
public void setLoadType(String loadType) {
    this.loadType = loadType;
}
public Integer getPrice() {
    return price;
}
public void setPrice(Integer price) {
    this.price = price;
}
public String getNotes() {
    return notes;
}
public void setNotes(String notes) {
    this.notes = notes;
}
public Organization getClient() {
    return client;
}
public void setClient(Organization client) {
    this.client = client;
```

```
    }
    public Dispatcher getDispatcher() {
        return dispatcher;
    }
    public void setDispatcher(Dispatcher dispatcher) {
        this.dispatcher = dispatcher;
    }
    public Driver getDriver() {
        return driver;
    }
    public void setDriver(Driver driver) {
        this.driver = driver;
    }
    public Truck getTruck() {
        return truck;
    }
    public void setTruck(Truck truck) {
        this.truck = truck;
    }
    public Trailer getTrailer() {
        return trailer;
    }
    public void setTrailer(Trailer trailer) {
        this.trailer = trailer;
    }
    public List<Stop> getStopList() {
        return stopList;
    }
    public void setStopList(List<Stop> stopList) {
        this.stopList = stopList;
    }
    public List<Cargo> getCargoList() {
        return cargoList;
    }
    public void setCargoList(List<Cargo> cargoList) {
        this.cargoList = cargoList;
    }
}
```

ДОДАТОК Д – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРСЬКОГО ВІДДІЛУ ЛОГІСТИЧНОГО ПІДПРИЄМСТВА «МІСТЕКСПРЕС»

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Програмне забезпечення для автоматизації роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

1.2 Програмне забезпечення, яке необхідне для функціонування програми

Для розгортання програмного забезпечення на сервері, який здатен опрацьовувати 50 активних користувачів, необхідні два комп’ютери . Сервер веб-застосунку повинен мати такі мінімальні характеристики:

– 16-ти поточний процесор;

– тактова частота процесора 3 ГГц;

– 16 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

Сервер бази даних повинен мати такі мінімальні характеристики:

– 4-х поточний процесор;

– тактова частота процесора 2.5 ГГц;

– 8 Гб оперативної пам’яті;

– 500 Гб вільного дискового простору.

1.3 Мови програмування

Для розробки серверної частини була використана мова Java, для розробки клієнтської частини класичні веб-технології HTML, CSS, JavaScript. Для зберігання даних була використана реляційна база даних PostgreSQL.

2 Функціональне призначення

2.1 Класи розв’язування задач та їх призначення

Функціональне призначення програмного забезпечення – автоматизація роботи диспетчерського відділу логістичного підприємства «МістЕкспрес».

Функції програмного застосунку, які доступні користувачам:

– контроль руху вантажівок в режимі реального часу;

– робота з замовленнями на перевезення;

– робота з рейсами;

– перегляд дошки зайнятості;

– робота з організаціями контрагентами;

– авторизація в системі;

– робота з активами підприємства;

– управління правами доступу користувачів до системи;

– отримання звітів;

– робота водія з призначеними рейсами.

2.2 Функціональні обмеження

Повинна бути виключена можливість несанкціонованого доступу до інформації та самої системи.

3 Опис логічної структури

3.1 Структура програми

Веб-застосунок побудовано на класичній для клієнт-серверних веб-додатків трьох шаровій архітектурі «Модель-представлення-контролер». Цей шаблон передбачає поділ програми на три взаємопов’язані частини:

- модель – основна компонент архітектури, що керує даними та містить в собі бізнес-логіку програми;
- представлення – інтерфейс користувача для вводу та отримання даних;
- контролер – компонент, що передає запити від представлення до моделі та віддає відповідь від моделі до представлення.

Перевагами даної архітектури є більша впорядкованість і зрозумілість структури коду завдяки жорсткому поділу коду на компоненти. Також ці властивості допоможуть полегшити внесення змін та розширення програми в майбутньому.

3.2 Алгоритми програми

Програма виконується на сервері, користувач має доступ до програми через інтерфейс веб-браузера.

3.3 Використовувані методи

Для розробки серверної частини використана технологія Java Enterprice Edition, для розробки клієнтської частини веб-фреймворк AngularJS. Для зберігання даних використана реляційна база даних PostgreSQL.

4 Технічні засоби, що використовуються

Для роботи програмного забезпечення необхідна наявність у диспетчера та адміністратора персонального комп’ютера, який має такі мінімальні характеристики:

– 8 Гб оперативної пам’яті;

– 150 Гб вільного дискового простору;

– інтегрована в процесор відеокарта;

– дисплей розміром 1536x864 пікселів;

– клавіатура та миша.

Для роботи програмного забезпечення необхідна наявність у водія смартфону.

5 Виклик та завантаження

Завантаження програмного забезпечення на сервер виконується шляхом розгортання .war архіву в контейнері сервлетів Tomcat.

Перехід користувача на сторінку веб-застосунку здійснюється за допомогою пошукової строки браузера.

6 Вхідні дані

Вхідними даними для роботи користувача з програмним забезпеченням є текстова інформація, що вводиться в поля вводу та форми на веб сторінках браузерного клієнта веб-застосунку та зберігається до бази даних.

7 Вихідні дані

Вихідні дані відображаються користувачу на веб-сторінках браузера, також вихідними даними є супровідна документація та звіти, що зберігаються на диск.