

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра «Комп'ютерні системи та мережі»
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему Розробка транспортної логістичної вебплатформи з використанням React.js

Виконав: студент 4 курсу, групи КНТз-519
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

БАЛАБУХА О.М.

(ПРІЗВИЩЕ та ініціали)

Керівник КИРИЧЕК Г.Г.

(ПРІЗВИЩЕ та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти бакалавр

Спеціальність 123 «Комп'ютерна інженерія»

(код найменування)

Освітня програма (спеціалізація) «Комп'ютерна інженерія»

(нова назва освітньої програми(спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«_____» _____ 2023 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БАЛАБУХА Олексій Миколайович

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проекту (роботи) Розробка транспортної логістичної вебплатформи з використанням React.js

керівник проекту (роботи) КИРИЧЕК Галина Григорівна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «21» березня 2023 року № 71

2. Строк подання студентом проекту (роботи) 10 травня 2023 року

3. Вихідні дані до проекту (роботи) дані ресурсів мережі інтернет, літературні та електронні джерела, перелік використовуваних програмних засобів: VS Code, JavaScript, TypeScript, React, Redux, WebSocket, Ant-design, Mapbox, Figma, Jest, React-dnd, Recharts

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз транспортної логістичної системи;

2) Розробка структури та алгоритму роботи системи логістичної вебплатформи;

3) Реалізація, дослідження та тестування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Алгоритм роботи процесу замовлення, Схема сторінок системи та доступ по ролям до них, Дизайну сторінок вебплатформи, Статистика системи

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	КИРИЧЕК Г. Г., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Робота з літературою	23.02.2023	
2	Розробка робочого плану роботи за узгодженням з науковим керівником	09.03.2023	
3	Написання 1 розділу та представлення його керівнику	15.04.2023	
4	Написання 2 розділу та представлення його керівнику	25.04.2023	
5	Написання 3 розділу та представлення його керівнику	05.05.2023	
6	Графічний матеріал та оформлення ПЗ	10.05.2023	
7	Загальне редагування та друк пояснювальної записки, графічного матеріалу	15.05.2023	
8	Проходження нормоконтролю	17.05.2023	
9			

Студент(ка)



(підпис)

Олексій БАЛАБУХА

(прізвище та ініціали)

Керівник проекту (роботи)



(підпис)

Галина КИРИЧЕК

(прізвище та ініціали)

РЕФЕРАТ

ПЗ: 78с; 6 табл; 22 рис; 3 додатки; 24 джерела.

БІБЛІОТЕКА, ВЕБПЛАТФОРМА, ДОДАТОК, ЗАПИТ, ЗАМОВЛЕННЯ, РОЛЬ, ТЕХНОЛОГІЯ, ТРАНСПОРТНА ЛОГІСТИКА

Об'єкт дослідження – система транспортної логістичної вебплатформи, створення та відстежування життєвого циклу замовлення.

Мета дипломної роботи - розробити швидку та зручну систему для створення замовлення для транспортування країною через зрозумілий інтерфейс веб-додатка з використанням бібліотеки, написаної на мові програмування JavaScript, React.js.

Проект складається з трьох розділів.

У першому розділі описується роль логістики у наші часи, її особливості та користь. Проведено аналіз усіх технологій, які використовуються у розробці даного додатку.

Другий розділ присвячено визначенню візуальних та функціональних вимог до вебплатформи. Також установлені етапи розробки сайту, архітектури і побудування структури проекту.

Третій розділ присвячено безпосередньо розробці дизайну та написання функціоналу додатку. В кінці розділу проведено тестування логістичної вебплатформи.

В результаті виконання дипломної роботи розроблено додаток, який не поступається іншим аналогам своїм простим і водночас корисним та зрозумілим функціоналом.

ЗМІСТ

Вступ.....	6
1 Аналіз транспортної логістичної системи	7
1.1 Основні поняття та визначення транспортної логістики	7
1.2 Огляд існуючих транспортних логістичних систем	14
1.3 Вибір React.js та додаткових бібліотек	21
1.4 Висновки до розділу 1	24
2 Розробка структури та алгоритму роботи системи логістичної вебплатформи	25
2.1 Етапи розробки транспортної логістичної вебплатформи.....	25
2.2 Вимоги до платформи розробки	29
2.3 Аналіз вимог до функціональності та інтерфейсу транспортної логістичної вебплатформи	32
2.4 Розробка архітектури системи	37
2.5 Висновки до розділу 2	43
3 Реалізація, дослідження та тестування системи	43
3.1 Розробка дизайну за допомогою онлайн-інструменту Figma.....	43
3.2 Розробка функціоналу за допомогою React.js та Nest.js	45
3.3 Аналіз результатів тестування	60
Висновки	63
Перелік джерел посилання	64
Додаток А	67
Додаток Б.....	68
Додаток В	72

ВСТУП

У сучасному світі транспортна логістика є важливою складовою бізнесу. Передові компанії, які працюють у цій сфері, стикаються з потребою у швидкому і ефективному виконанні доставки товарів. Для цього потрібні не тільки надійні транспортні засоби, але й комплексні рішення в області логістики.

Такі комплексні рішення можуть бути надані за допомогою транспортної логістичної веб платформи. Така платформа забезпечує зв'язок між замовниками і виконавцями, що дає змогу здійснювати транспортні перевезення в найбільш ефективний спосіб.

Одним з ключових аспектів розробки транспортної логістичної веб платформи є використання сучасних технологій. В даній роботі ми розглянемо можливість використання бібліотеки React.js для розробки клієнтської частини веб платформи.

React.js - це бібліотека JavaScript для створення користувацьких інтерфейсів. Вона дає змогу створювати ефективні інтерактивні інтерфейси, забезпечуючи швидкий рендеринг та мінімальну кількість оновлень сторінки. Використання React.js для розробки транспортної логістичної веб платформи дозволить забезпечити її ефективність та зручність використання для користувачів.

Метою дипломної роботи є розробка транспортної логістичної веб-платформи з використанням React.js, що має на меті поліпшення ефективності транспортних процесів і забезпечення зручного та швидкого доступу до логістичної інформації для користувачів. Робота спрямована на дослідження і аналіз потреб ринку в транспортних логістичних рішеннях, визначення функціональних вимог до веб-платформи та розробку програмного забезпечення на базі React.js з використанням сучасних підходів до розробки веб-додатків. Основним результатом роботи є функціональна та ефективна транспортна логістична веб-платформа, яка відповідає потребам користувачів і допомагає оптимізувати транспортні процеси.

1 АНАЛІЗ ТРАНСПОРТНОЇ ЛОГІСТИЧНОЇ СИСТЕМИ

1.1 Основні поняття та визначення транспортної логістики

Глобалізаційні процеси, які відбуваються в усьому світі, впливають на розвиток цивілізації, адже відкриваються кордони, які раніше були закритими. Тому важливим стає розвиток логістики, яка, в свою, чергу також впливає на глобалізацію та відіграє велику роль у цьому процесі.

Під глобалізацією світового господарства розуміють процес посилення взаємозв'язку національних економік країн світу, що знаходить своє відображення в утворенні світового ринку товарів і послуг, фінансів; становленні глобального інформаційного простору, перетворенні знання в основний елемент суспільного багатства, виході бізнесу за національні кордони через формування транснаціональних корпорацій, впровадженні й домінуванні в повсякденній практиці міжнародних відносин і внутрішньополітичного життя народів принципово нових, універсальних ліберально демократичних цінностей тощо [12].

Дослідженню загальної логістики або окремих її напрямів присвятили свої наукові праці такі відомі вітчизняні та зарубіжні вчені як Гаджинський А. М., Горбенко О.В., Крикавський Є. В., Барді Е., Бест М., Вордлоу Д., Джеймс Р. Сток, Дуглас М. Ламберт, Дж. Шапіро, Дональд Бауресокс Джонсон Д. та інші.

Вивчення логістики вважається вкрай важливим у сучасному світі. Підприємства залежать від професіоналів у сфері логістики, щоб підтримувати виробництво та доставку. Вивчення логістики є першим кроком до задоволення цієї критичної бізнес-потреби та створення основи для перспективної кар'єри. Вивчення цього питання настільки важливе, бо без логістики світ зупинився б.

Метою логістики є своєчасне та економічно найефективне задоволення всіх вимог замовників.

Багато компаній спеціалізуються на логістиці, надаючи послуги виробникам, роздрібним торговцям та іншим галузям, які мають велику потребу в транспортуванні товарів.

Деякі володіють усією інфраструктурою, від реактивних літаків до вантажівок, складів і програмного забезпечення, а інші спеціалізуються на одній або двох частинах. FedEx, UPS і DHL є відомими постачальниками логістичних послуг.

Говорячи простою мовою, мета управління логістикою полягає в тому, щоб мати потрібну кількість ресурсу або вхідних ресурсів у потрібний час, доставити їх у відповідне місце в належному стані та доставити правильному внутрішньому чи зовнішньому клієнту [10].

Поняття «логістика» походить від французького слова «Loger», що означає мистецтво основного навчання до руху і поставку товарів. Це стратегія або концепція, що зазвичай застосовується до управління маркетингом [9].

Логістика – це планування, організація і контроль усіх видів діяльності по переміщенню і складуванню, що забезпечують проходження матеріальних і зв'язаних з ними інформаційних потоків від пункту закупівлі сировини до пункту кінцевого споживання.

Найбільш поширеним і змістовним можна вважати визначення логістики як планування, організацію та управління рухом матеріальних, інформаційних та фінансових ресурсів в різних соціально-економічних системах. В вітчизняній науці та практиці сьогодні логістика в більшій мірі розглядається як організація раціонального руху товарів та послуг від постачальника до споживача з метою ефективної організації і управління матеріальними і інформаційними потоками [8].

Впровадження передового досвіду логістики став одним з найбільш цікавих і складних оперативних областях бізнесу та управління державним сектором. Логістика є унікальним явищем, яке ніколи не зупиняється. Логістика відбувається в усьому світі 24 години на добу, сім днів на тиждень, протягом 52 тижнів на рік [9].

Транспортування та складування є двома основними функціями логістики. Управління транспортуванням зосереджено на плануванні, оптимізації та виконанні використання транспортних засобів для переміщення товарів між

складами, торговими точками та клієнтами. Транспорт є мультимодальним і може включати морський, повітряний, залізничний та автомобільний транспорт [10].

Транспорт як галузь матеріального виробництва відноситься до сфери матеріальних послуг. Він включає в себе не лише транспортні засоби різних видів та типів, а й транспортну інфраструктуру та підготовлений персонал. Значна частина логістичних операцій в русі матеріального потоку від джерела сировини до кінцевого споживання здійснюється із застосуванням транспорту.

Треба зазначити, що метою транспортної логістики є якісне просування необхідних матеріальних потоків до споживача відповідно до графіка у встановлений час, з мінімальними витратами і належним рівнем сервісу.

Транспортна логістика являє собою процес управління вантажопотоками (пасажиропотоками) і супутніми потоками (інформаційними, фінансовими, сервісними, матеріальними для власних нестатків і ін.).

Важливим є те, що функції логістики не починаються на станції відправлення і не завершуються на станції призначення. Вони починаються з визначення пропозиції, необхідного для освоєння необхідних обсягів перевезень, і продовжуються на всіх етапах обслуговування споживачів.

Сутність поняття транспортної логістики дозволяє означити межі цієї функціональної сфери. Вивченню транспортної логістики в свою чергу приділялось багато уваги з боку різних науковців, таких як А. С. Маловичко, І. Г. Смірнов, Р. Р. Галяутдинов, Т. І. Савенкова, Б. А. Анікін, В. Г. Банько, А. М. Гаджинський та інші, тож значна кількість наукових праць призначено саме цьому поняттю.

А. М. Гаджинський розглядає транспортну логістику як значну частину логістичних операцій на шляху руху матеріального потоку від постачальника до кінцевого споживача, що здійснюються транспортними засобами.

Н.В. Чепель визначає транспортну логістику як оптимізацію транспортних систем, вибір виду і типу транспортних засобів; визначення різноканальних маршрутів доставки; забезпечення технологічної єдності транспортно- складського процесу [1].

Транспортна логістика визначається як сфера діяльності, що охоплює три області:

- процес планування, організації і здійснення раціональної і недорогої доставки (перевезення) вантажів (товарів) від місць їх виробництва і до місць споживання;
- контроль за всіма транспортними і іншими операціями, що виникають на шляху доставки вантажів з використанням сучасних засобів телекомунікації, інформатики і інших інформаційних технологій;
- надання відповідної інформації вантажовласникам [12].

Однак варто зазначити, що на сьогодні немає єдиного чітко прийнятого визначення, тому що думки науковців з приводу поняття «транспортної логістики» розділились, що свідчить про певне розмаїття поглядів на цей рахунок, тож нижче наведемо основні поняття транспортної логістики (таблиця 1.1).

Таблиця 1.1 - Визначення «транспортна логістика» різними науковцями

Автор	Визначення
1	2
Маловичко А. С. [2]	Транспортна логістика являє собою процес організації ланцюга поставки та управління цим ланцюгом. Цей ланцюг охоплює постачання сировини, необхідної для виробництва, управління матеріальними ресурсами на підприємстві, постачання готової продукції на склади та розподільчі центри, сортування, переробку, пакування, розподіл продукції у містах споживання.
Тридід О. М. [3]	Транспортна логістика – це система з організації доставки, а саме з переміщення будь-яких матеріальних предметів, речовин тощо з однієї точки в іншу за оптимальним маршрутом.
Смірнов І. Г. [4]	Транспортна логістика – управління фізичним переміщенням матеріальних ресурсів відповідно до інтересів їх споживачів у просторі та часі.
Сокур І.М. [5]	Транспортна логістика – це вид логістики, що управляє комплексом операцій, які забезпечують фізичне переміщення товарно-матеріальних цінностей між учасниками ланцюга поставок з мінімальними витратами, тобто переміщення потрібної кількості товару в необхідну точку, оптимальним маршрутом за необхідний час і з найменшими витратами.

Автор	Визначення
1	2
Пономарьова Ю. В. [6]	Транспортна логістика вирішує комплекс задач, пов'язаних з організацією переміщення вантажів транспортом загального користування.

Більш розлогим є визначення автора Маловичко А.С., відповідно до якого транспортна логістика виконує функцію управління матеріальними ресурсами на підприємстві, сортування, переробку, пакування, розподіл продукції у містах споживання.

У роботі автора Н.В. Чепель також йдеться про те, що транспортна логістика – це також оптимізація транспортних систем, вибір виду і типу транспортних засобів; визначення різноканальних маршрутів доставки; забезпечення технологічної єдності транспортно- складського процесу.

Розмежування на області є у визначенні автора Дроздова П.А., а саме: процес планування, контроль за всіма транспортними та іншими операціями, надання інформації вантажовласникам.

У роботі автора Сокура І.М. також зауважується, що переміщення товарно-матеріальних цінностей між учасниками ланцюга поставок здійснюється з мінімальними витратами.

Важливим є те, що жоден з авторів у запропонованих визначеннях не враховує, що транспортна логістика може стосуватися не тільки переміщення вантажів, але й пасажирів.

Отже, можна запропонувати наступне визначення, сформоване Лифар В. В.: транспортна логістика – це функціональна область логістики, пов'язана з управлінням процесами переміщення потоків вантажів і пасажирів необхідної кількості, у необхідне місце, оптимальним маршрутом, за необхідний час, з найменшими витратами та відповідним інформаційним забезпеченням [7].

Досліджуючи транспортну логістику, слід зазначити, що найважливішим питанням є саме гарантія безпеки доставки кінцевим споживачам.

Транспортна логістика також є важливою складовою економіки через інтенсивне використання інфраструктури. Мобільність людей та вантажів, а також

інформаційні та комунікаційні технології з кожним роком грають все більшу роль у глобальній економіці та є індикаторами економічних можливостей тої чи іншої країни.

Транспортна логістика вирішує комплекс завдань, пов'язаних з організацією переміщення вантажів транспортом загального користування. Основними з цих завдань є:

- вибір виду транспортного засобу;
- вибір типу транспортного засобу;
- оптимізація транспортного процесу під час змішаних перевезень;
- визначення раціональних маршрутів доставки;
- забезпечення технологічної єдності транспортно-складського процесу;
- координація транспортного і виробничого процесу.

Роль транспорту істотно змінюється з розвитком логістичних систем. У сучасних умовах дисципліна транспортного обслуговування визначається не інтересами окремого відправника (одержувача), а оптимальним співвідношенням витрат і прибутку в зазначеному циклі виробництва і споживання.

Існують такі ознаки класифікації транспортної складової логістичних систем:

а) за видом доставки:

- 1) пряма;
- 2) з переробкою на транспортних терміналах;
- 3) з переробкою і зберіганням у розподільчих центрах.

б) за видом обслуговування:

- 1) зі складу постачальника або розподільчого центру на склад споживача або розподільчий центр;
- 2) зі складу постачальника або розподільчого центру безпосереднього споживачу;
- 3) з виробництва постачальника у виробництво споживача без складського зберігання та переробки.

в) за видами транспортного сполучення:

- 1) пряме;

2) змішане.

Будучи галуззю матеріального виробництва, транспорт має свою продукцію – це сам процес переміщення, яка характеризується рядом істотних відмінностей:

- відсутність речової форми, але в той же час матеріальність за своїм характером, тому що в процесі переміщення затрачаються матеріальні засоби: відбувається зношення рухомого складу і засобів обслуговування, використовується праця робітників транспортної сфери і т.д.;

- неможливість зберігання і нагромадження, тому транспорт може мати тільки деякий резерв своєї пропускної та провідної здатності для задоволення потреб у транспортних послугах;

- втілення в додаткових транспортних витратах, які зв'язані з переміщення матеріального потоку, тому транспорт необхідно використовувати так, щоб транспортні витрати були найменшими за інших рівних умов;

- прив'язаність до певного місця, району, регіону (наприклад, до місця, де розташовані шляхи сполучення і є відповідні транспортні підприємства) [11].

Вказані особливості накладають свій відбиток на устрій і функціонування системи транспортної логістики.

Виділяють такі функції транспортної логістики:

- системоутворююча функція. Логістика є системою ефективних технологій забезпечення процесу управління ресурсами. Логістика утворює систему управління товарорухом, тобто формуванням господарських зв'язків, організацією пересування продукції через місця складування, формування і регулювання запасів продукції, розвиток та організація складського господарства;

- інтегруюча функція. Логістика забезпечує синхронізацію процесів збуту, зберігання і доставки продукції з орієнтацією їх на ринок засобів виробництва і надання посередницьких послуг споживачам. Забезпечує узгодження інтересів логістичних посередників у логістичній системі;

- регулююча функція. Логістичне управління матеріальними потоками спрямоване на економію всіх видів ресурсів, скорочення витрат живої й упредметненої праці на стику різних організаційно-економічних рівнів і галузей.

Управлінський вплив полягає в підтримці відповідності поведінки частини логістичної системи інтересам цілого;

- результуюча функція. Логістична діяльність спрямована на постачання продукції в необхідній кількості, у зазначений час і місце з заданою якістю, за мінімальних витрат. Логістика прагне охопити всі етапи взаємодії ланцюга «постачання-виробництво-розподіл-споживання», вона є алгоритмом перетворення ресурсів у постачання готової продукції відповідно до існуючого попиту.

Головна мета транспортної логістики конкретизується в її завданнях, які за ступенем значимості можна розділити на три групи: глобальні, загальні, часткові (локальні) [12].

Таким чином, розглянувши основні поняття та визначення «транспортна логістика» можна стверджувати, що транспортна логістика є процесом організації ланцюга поставки, а також оптимізації транспортних систем, вибір виду і типу транспортних засобів, що виконуються за допомогою процесу планування, контролю за всіма транспортними та іншими операціями, наданням відповідної інформації вантажовласникам та здійснюється з мінімальними витратами.

Транспортна логістика також є важливою складовою економіки через інтенсивне використання інфраструктури. Мобільність людей та вантажів, а також інформаційні та комунікаційні технології з кожним роком грають все більшу роль у глобальній економіці.

1.2 Огляд існуючих транспортних логістичних систем

Транспортно-логістичні системи (ТЛС) є одними з найважливіших складових, які забезпечують економічний розвиток підприємств, регіонів і України в цілому. Вони є тією ланкою, яка зв'язує виробників, торгівлю та споживачів між собою. Від рівня транспортно-логістичного обслуговування залежить рівень

задоволення потреб споживачів у якісних товарах та послугах. Особливої уваги заслуговує розвиток транспортно-логістичних систем в умовах інтеграції України у європейську та світову економічну систему.

Проблемам розвитку транспортно-логістичних систем присвячені роботи як зарубіжних, так і вітчизняних вчених. Серед них роботи В.В. В. Андерсон, Дж. К. Вілла [13] Брагінського [14], В.Л. Диканя [15], О.М. Дороховського [16], Є.В. Крикавського [17], І.Г. Лебідь та В.В. Крикуна [18], Т. Сирийчика [19], О. Є. Соколова [22] та ін.

Дослідження науковців стосуються розвитку різних видів транспорту, управлінської науки, маркетингового підходу та інших актуальних питань, пов'язаних з транспортно-логістичною системою або її складовими.

Логістичні системи вкладаються у загальноприйняте поняття «системи», бо складаються із системоутворюючих елементів, тісно взаємопов'язаних і взаємозалежних між собою, які мають впорядковані зв'язки й утворюють певну структуру із заздалегідь заданими властивостями. Відрізняються ці системи високим ступенем узгодженості вхідних продуктивних сил з метою управління наскрізними матеріальними потоками [20].

Тож першочергово необхідно надати визначення поняттю «система».

Система – це сукупність функціонально впорядкованих підсистем та елементів, що знаходяться у відносинах і зв'язках один з одним, утворюють певну закономірну цілісність, єдність, і визначають її призначення і спрямованість на досягнення заданої мети (Апопій та ін., 2005).

Логістична система (ЛС) — це адаптивна система зі зворотним зв'язком, яка виконує ті чи інші логістичні функції (операції), складається із підсистем і має розвинуті внутрішньосистемні зв'язки та зв'язки із зовнішнім середовищем [20].

Акцентуючи дещо більшу увагу на окремих властивостях системи, окремі вчені формують різні підходи до визначення логістичної системи.

Так, логістична система – адаптивна система із зворотним зв'язком, що виконує ті чи інші функції та операції, складається, переважно, з декількох підсистем і має досить розвинуті зв'язки із зовнішнім середовищем.

Логістична система – організаційноуправлінський механізм координації, який дає змогу досягти ефекту завдяки чіткій злагодженості у діях спеціалістів різноманітних служб, які беруть участь в управлінні матеріальним потоком [21].

Метою логістичної системи є забезпечення наявності необхідного товару в необхідній кількості та заданої якості в потрібному місці й у потрібний час для потрібного споживача із заданими витратами.

Будь-яка логістична система складається із сукупності елементів, так званих ланок логістичної системи, між якими встановлені певні функціональні зв'язки і відношення. Внутрішньосистемні зв'язки є більш міцними, ніж зв'язки із зовнішнім середовищем. Зазвичай вони мають циклічний характер, бо відображають послідовність передачі матеріального та інформаційного потоків між ланками відповідного логістичного ланцюга.

Соколова О. Є. розглядає транспортно-логістичну систему як інтегровану сукупність суб'єктів транспортно-логістичної діяльності та об'єктів транспортно-логістичної інфраструктури, що взаємодіють між собою з метою оптимізації руху вантажопотоків «від дверей до дверей» за мінімальними витратами на максимально вигідних умовах [22].

Наведемо нижче рис.1.1, що розкриє властивості та всі складові транспортно-логістичної системи.



Рисунок 1.1 - Транспортно-логістична система: властивості та складові

Таким чином, транспортно-логістична система множину взаємопов'язаних елементів у вигляді підприємств транспорту, логістики, транспортних мереж, які утворюють єдине ціле та підпорядковані одній меті (цільовій функції системи) - задоволення потреб споживачів у транспортних та логістичних послугах [23].

Логістичні системи класифікуються за такими основними ознаками. Принципово за ознакою просторового обмеження логістичні системи поділяються на два типи:

- макрологістичні;
- мікрологістичні.

Макрологістична система є великою логістичною системою управління поточними процесами за участю декількох і більше незалежних суб'єктів господарювання, не обмежених у територіальному розташуванні.

Виділяють такі макрологістичні системи:

- регіональні;
- національні (міжрегіональні);
- міжнаціональні.

Створення макрологістичних систем обумовлене необхідністю забезпечити чітку взаємодію різногалузевих структур з метою поліпшення економічного стану на глобальному рівні. Під час створення макрологістичних систем особливу увагу приділяють взаємопогодженню інтересів кожного учасника незалежно від його ролі у створюваній системі.

Цілі створення макрологістичних систем можуть значною мірою відрізнятися від цілей і критеріїв синтезу мікрологістичних систем. У більшості випадків критерій мінімуму загальних логістичних витрат використовується і під час синтезу макрологістичних систем. Однак найчастіше критерії формування макрологістичних систем визначаються екологічними, соціальними, військовими, політичними та іншими цілями. Наприклад, для поліпшення екологічної ситуації в регіоні може бути створена макрологістична система оптимізації транспортних (вантажних) регіональних потоків, що вирішує задачі оптимізації маршрутів, розв'язання транспортних потоків, переключення з одного виду транспорту на

інший. З погляду державних органів управління, які також можуть брати участь у створенні макрологістичної системи, позитивний ефект може виражатися, наприклад, у поліпшенні загальної економічної ситуації в регіоні, країні або між державами.

Мікрологістичну систему підприємства можна подати у вигляді основних підсистем: закупівлі, виробництва і збуту (рис. 1.2).

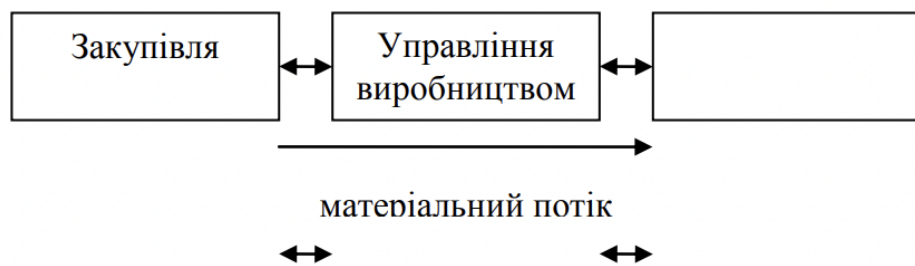


Рисунок 1.2 - Мікрологістична система

Мікрологістичну систему підприємства можна представити у більш деталізованому вигляді (рис. 1.3).

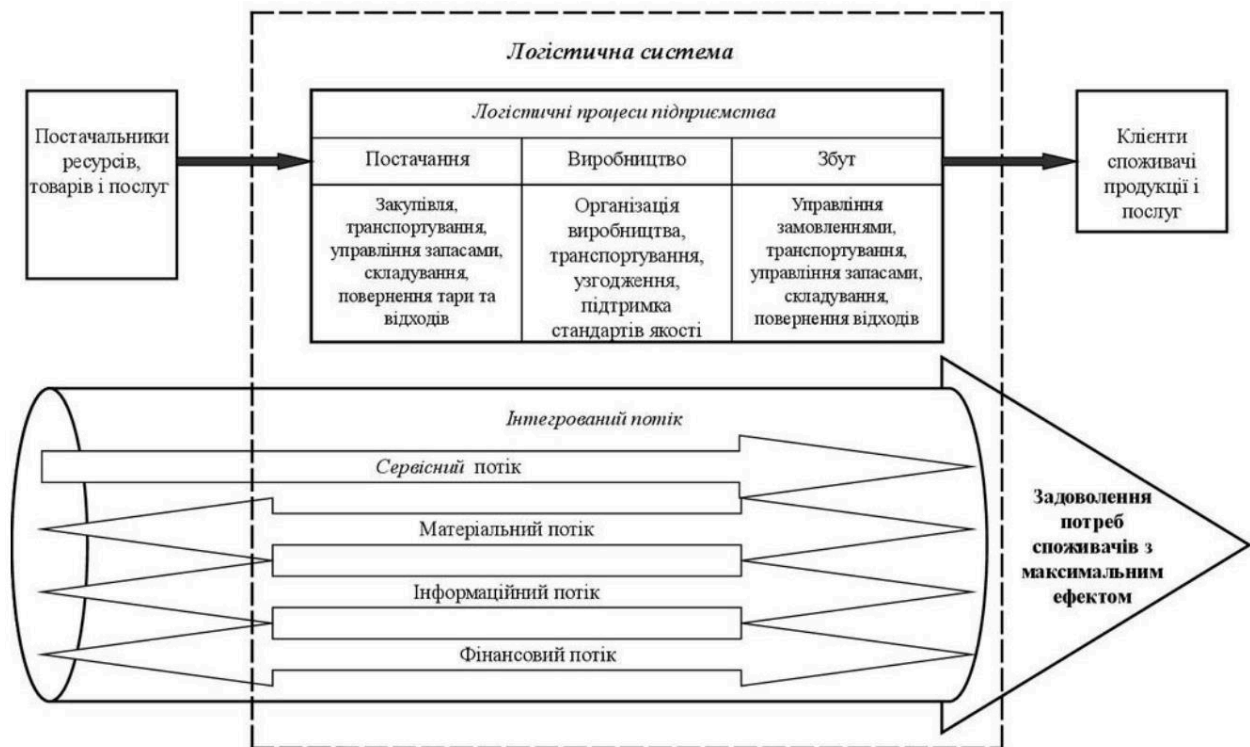


Рисунок 1.3 - Мікрологістична система у деталізованому вигляді

Таким чином, макрологістична система є високоінтегрованою інфраструктурою економіки регіону, країни або групи країн.

Мікрологістична система охоплює внутрішньовиробничу логістичну сферу одного підприємства або групи підприємств, об'єднаних на корпоративних засадах. До мікрологістичної системи належать технологічно пов'язані виробництва, об'єднані єдиною інфраструктурою, які працюють на єдиний економічний результат.

Залежно від виду логістичних ланцюгів логістичні системи можуть бути:

- логістичні системи з прямими зв'язками - це системи, у яких матеріальний потік доводиться до споживача без посередників, на основі прямих господарських зв'язків;

- ешелоновані (багаторівневі) логістичні системи - це системи, у яких матеріальний потік доводиться до споживача за участю як мінімум одного посередника.

- гнучкі логістичні системи - системи, у яких доведення матеріального потоку до споживача здійснюється як за прямими зв'язками, так і за участю посередників.

Враховуючи багатоаспектність структури логістичних систем, різноплщинність їхнього функціонування та багатокритеріальність управління ними, найповніше виглядає класифікація і структуризація логістичних систем, подана нижче (табл. 1.2) [24].

Таблиця 1.2 - Класифікація і структуризація логістичних систем

1	2
Інституціональний поділ	Мікрологістична система Металогістична система (логістичний ланцюг) Мезологістична система Макрологістична система Зовнішня логістична система (міжсистема)
Фазовий поділ	Логістична підсистема у сфері постачання Логістична підсистема у сфері виробництва Логістична підсистема у сфері збуту Логістична підсистема у сфері повернень товарів Інтегрована підсистема матеріальної логістики

Продовження таблиці 1.2

1	2
Фазовий поділ	Інтегрована підсистема маркетингової логістики Інтегрована логістична підсистема постачальників Інтегрована логістична підсистема споживачів Інтегрована логістична підсистема у сфері торгівлі
Функціональний поділ	Логістична підсистема транспортування Логістична підсистема формування запасів Логістична підсистема складування Логістична підсистема пакування
Поділ за функціями управління	Підсистема логістичного планування Підсистема логістичного керування Підсистема організації логістики Підсистема логістичного контролювання Підсистема нормативного логістичного управління Підсистема стратегічного логістичного управління Підсистема операційного логістичного управління Підсистема інтегрованого логістичного управління
Предметно-структурний поділ	Підсистема інтегрованих переміщень товарів (фізичні потоки) Підсистема інтегрованих інформації та управлінських рішень щодо переміщень (інформаційні потоки) Підсистема регуляції та страхування (у вартісному сенсі) логістичних рішень і процесів (фінансові потоки)
Поділ за компонентами ефективності	Підсистема логістичних витрат Підсистема послуг і логістичного обслуговування

Таким чином, був проведений огляд макрологістичних та мікрологістичних транспортно - логістичних систем та, враховуючи багатоаспектність структури логістичних систем, різноплщинність їхнього функціонування та багатокритеріальність управління ними, надана класифікація і структуризація логістичних систем. Було встановлено, що існують логістичні системи з прямими зв'язками, ешелоновані (багаторівневі), а також гнучкі.

1.3 Вибір React.js та додаткових бібліотек

Для розробки додатку використовується бібліотека React.js, яка розроблена на мові програмування JavaScript, одному з найпоширеніших мов програмування на сьогоднішній день.

Javascript - це мова програмування, яку використовують для створення інтерактивних веб-сторінок. JavaScript було створено для того, щоб “оживити вебсторінки”. Програми цією мовою називаються скриптами. Їх можна писати прямо на сторінці в коді HTML і вони автоматично виконуватимуться при завантаженні сторінки. Скрипти надаються та виконуються як простий текст. Для запуску їм не потрібна спеціальна підготовка чи компілятор.

Для простої роботи з кодом корисно використовувати фреймворки, вони мають під копитом безліч функцій та методів, які можуть полегшити розробку веб-додатку. Існує безліч фреймворків, які розроблені на базі JavaScript: Angular.js, Vue.js, Ember.js, тд. Але мною обрано бібліотеку React.js.

React.js - це бібліотека для створення користувацьких інтерфейсів. Вона дозволяє розробникам створювати динамічні веб-додатки, використовуючи компонентний підхід, який спрощує роботу зі станом додатка та рендерингом компонентів на стороні клієнта. Головним плюсом цієї бібліотеки є, то що ми пишемо додаток в декларативному стилі. Нам потрібно описувати як наші компоненти будуть вести себе в певних станах.

Redux - це популярна бібліотека управління станом додатка в Redux-архітектурі. Вона забезпечує зручний спосіб організації стану додатка, дозволяє забезпечити централізоване управління станом, спрощує взаємодію між компонентами та допомагає уникнути проблем з передачею стану через вкладені компоненти(рис. 1.4).

TypeScript - це сучасний мовний розширення для JavaScript, який додає статичний типізації, забезпечуючи більшу надійність, підвищену продуктивність та зручність розробки. Використання TypeScript у поєднанні з React.js дозволяє

забезпечити більшу безпеку коду, запобігти потенційним помилкам та полегшити роботу зі складними даними.

WebSocket - це технологія двостороннього зв'язку між веб-браузером та веб-сервером в режимі реального часу. Використання WebSocket дозволяє взаємодія між клієнтом та сервером без необхідності постійно оновлювати сторінку, що дозволяє створювати додатки з реальним часом оновлення даних.

Ant-design – це одна з найпопулярніших бібліотек дизайну користувацького інтерфейсу (UI) в екосистемі React. Вона надає набір готових компонентів, стилів та інструментів, які допомагають розробникам швидко створювати сучасні та привабливі веб-додатки з високою якістю користувацького інтерфейсу.

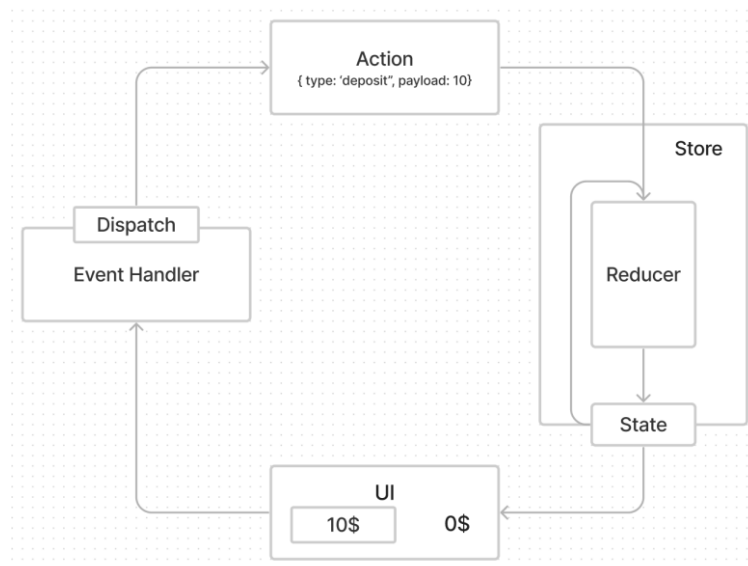


Рисунок 1.4 - React-Redux архітектура

Плюси ant-design:

- швидка розробка: готові компоненти: Ant Design надає велику кількість готових компонентів, таких як кнопки, форми, таблиці, модальні вікна, пагінація та багато інших. Вони мають сучасний вигляд, підтримують різні функції та можуть бути легко налаштовані за допомогою властивостей;
- гнучкість та налаштовуваність: Ant Design надає багатий набір налаштувань та параметрів, що дозволяють розробникам легко налаштувати вигляд

компонентів під свої потреби, такі як зміна кольорів, типографії, розташування тощо;

– кросс-браузерна сумісність: Ant Design забезпечує кросс-браузерну сумісність, що дозволяє розробляти веб-додатки, які працюють однаково добре на різних веб-браузерах, таких як Chrome, Firefox, Safari, та інші.

Загалом, Ant Design - це потужна бібліотека, яка надає розробникам можливість швидко створювати сучасні веб-додатки з професійним дизайном. Завдяки цим особливостям, Ant Design забезпечує ефективний процес розробки веб-додатків та допомагає забезпечити професійний вигляд та високу якість користувацького інтерфейсу(рис 1.5).

Mapbox - це високопрофільна бібліотека для роботи з картографією та геопросторовими даними. Вона надає можливості рендерингу інтерактивних карт, роботи з геоданими, взаємодії з користувачем на карті та інші функції, що роблять її ідеальним вибором для розробки веб-додатків з географічними функціями.

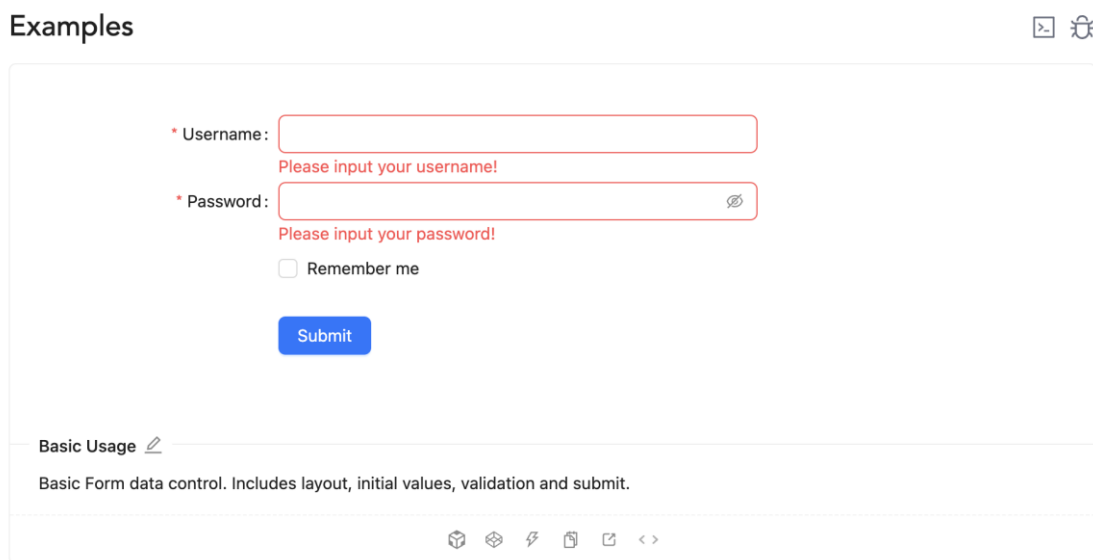


Рисунок 1.5 - Форма з валідацією зі застосуванням ant-design

Figma - це онлайн-інструмент для дизайну і прототипування інтерфейсів, який дозволяє командам працювати разом в режимі реального часу з будь-якого місця з Інтернетом. Він забезпечує зручний інтерфейс для створення векторної графіки, макетів і прототипів, що можуть бути легко налаштовані для спільної

роботи та збережені в хмарі. Figma дозволяє дизайнерам створювати і редагувати макети, додавати взаємодію та анімацію, розробляти бібліотеки компонентів, спілкуватися з іншими учасниками проекту та зберігати всю інформацію в одному місці. Figma є популярним інструментом для дизайну інтерфейсів та розробки макетів веб-сайтів та мобільних додатків.

Вибір цих бібліотек та технологій обґрунтовано їх популярністю, гнучкістю, можливостями та відповідністю вимогам дипломної роботи. Вони дозволяють розробникам ефективно реалізувати поставлені завдання, забезпечують високу якість коду, зручність розробки та відповідають сучасним стандартам розробки веб-додатків.

1.4 Висновки до розділу 1

Таким чином, розглянувши основні поняття та визначення «транспортна логістика» можна стверджувати, що транспортна логістика є процесом організації ланцюга поставки, а також оптимізації транспортних систем, вибір виду і типу транспортних засобів, що виконуються за допомогою процесу планування, контролю за всіма транспортними та іншими операціями, наданням відповідної інформації вантажовласникам та здійснюється з мінімальними витратами.

Транспортна логістика також є важливою складовою економіки через інтенсивне використання інфраструктури. Мобільність людей та вантажів, а також інформаційні та комунікаційні технології з кожним роком грають все більшу роль у глобальній економіці.

Отже, був проведений огляд макрологістичних та мікрологістичних транспортно - логістичних систем та, враховуючи багатоаспектність структури логістичних систем, різноплщинність їхнього функціонування та багатокритеріальність управління ними, надана класифікація і структуризація

логістичних систем. Було встановлено, що існують логістичні системи з прямими зв'язками, ешелоновані (багаторівневі), а також гнучкі.

Вибір React.js та інших бібліотек та технологій обґрунтовано їх популярністю, гнучкістю, можливостями та відповідністю вимогам дипломної роботи. Вони дозволяють розробникам ефективно реалізувати поставлені завдання, забезпечують високу якість коду, зручність розробки та відповідають сучасним стандартам розробки веб-додатків.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМУ РОБОТИ СИСТЕМИ ЛОІСТИЧНОЇ ВЕБПЛАТФОРМИ

2.1 Етапи розробки транспортної логістичної вебплатформи

Перед тим як розпочати проектування нашого додатку потрібно зробити детальний план розробки вебплатформи. У цьому плані ми розглянемо всі етапи розробки від вибору бібліотек, за допомогою яких працюватиме додаток, до тестування. Це допоможе нічого не прогавити, та розробити платформу, яка працюватиме без будь-яких дефектів. Отже, нижче наведено перелік етапів, які ми розглянемо:

- вибір мови програмування та бібліотек;
- планування та аналіз вимог;
- проектування;
- розробка;
- тестування системи.

Бібліотеки та мова програмування вже обрані, тож перший етап вже завершено, це означає що можна переходити до наступного.

Наступний крок – це аналіз вимог до функціональності та інтерфейсу. Спочатку треба визначити функціональну складову нашого додатку, тобто

зрозуміти які завдання повинен вирішувати сайт, яку мету він має, яка цільова аудиторія, що очікується від його роботи та функціоналу, скільки сторінок матиме платформа, чи треба авторизація користувачів, вимоги до інтеграції мап, гугл сервісів, тощо. Усі ці аспекти треба визначити у першу чергу, бо на вимогах до функціоналу ґрунтується подальша наша робота. Існує безліч способів донести до розробника вимоги до функціоналу програми: таблиця, перелік, схема(Рис 2.1). Наша платформа матиме декілька сторінок, кожна міститиме свій функціонал, тож треба детально продумати як працюватиме кожна.

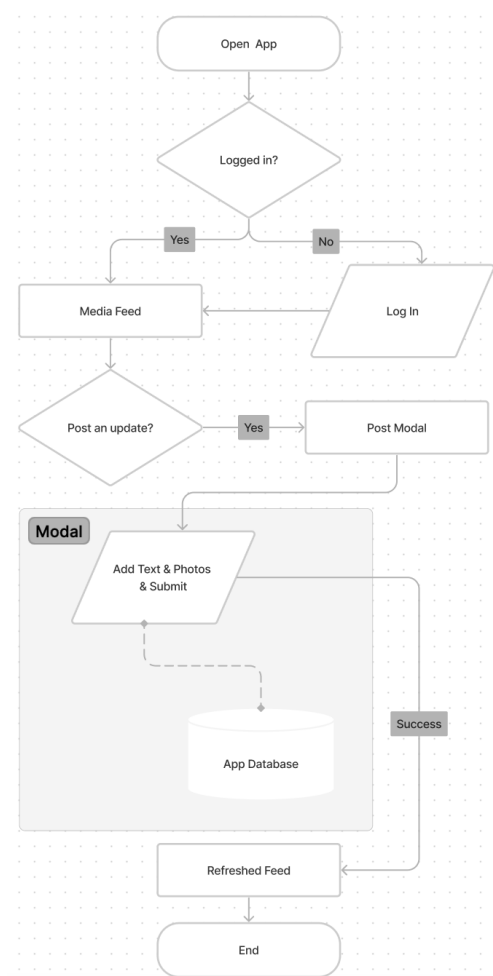


Рисунок 2.1 - Схема функціоналу авторизації, зміна тексту існуючого поста та збереження оновлення в базу даних

Після цього можна одразу приступати до розробки дизайну нашої вебплатформи. Потрібно з'ясувати як повинен виглядати наш додаток – який

контент буде на кожній сторінці, розташування елементів, кольори, анімація блоків і тд. Приклад переліку вимог до інтерфейсу:

- бренд-ідентичність - сайт треба зробити унікальним, зі своєю палітрою кольорів, логотипом, шрифтами та заголовками;
- навігація – навігація повинна бути простою та легкою в користуванні, дозволяючи відвідувачам знайти потрібну інформацію швидко та легко;
- мобільна оптимізація – існує безліч пристроїв з різним форматом екрану, тож треба оптимізувати розміри блоків, шрифти, тощо під кожний з них, щоб на комп'ютері, планшеті чи смартфоні сайт виглядав добре, без усяких дефектів;
- графіка та фотографія - графіка та фотографії повинні використовуватись для підсилення повідомлень та допомоги у візуальному сприйнятті. Зазвичай користувачів притягують програми з цікавою анімацією;
- швидкість завантаження - сайт повинен бути добре оптимізовано, щоб швидко завантажуватися аби уникнути розчарування відвідувачів та погіршення позицій сайту у пошукових системах.

Отже, функціонал веб-сайту слід визначати перед дизайном, оскільки він є його основною складовою та визначає його цільове призначення, а дизайн повинен бути заснований на функціональних вимогах, щоб відображати їх максимально точно та ефективно.

Проектування. Це наступний етап нашої розробки. На цьому етапі розглянемо зробити дві речі. Перше - одним із основних дій цього етапу, та і в цілому нашого плану, це розробка архітектури для frontend та backend частин. Вище написані визначення кожної технології, які використовуються у цій роботі, але саме в описі архітектури ми детально та послідовно розглянемо чому обрано саме ці бібліотеки, порівняння з іншими аналогами, та як саме треба використовувати кожну бібліотеку. Також налаштуємо маршрутизацію проекту, розроблено структуру, з детальним поясненням кожного файлу та директорії, що допоможе правильно розподілити компоненти у редакторі коду, функціональність та взаємозв'язки всіх елементів сайту(рис. 2.2). Це дасть змогу забезпечити ефективну та оптимізовану

роботу сайту, а також зробити його зручним та зрозумілим для користувачів. Створення архітектури розробки сайту також допомагає побудувати систему управління контентом та базу даних, яка використовуватиметься для зберігання та обробки інформації. Навіть невеликий сайт може мати складну структуру, тому важливо враховувати всі потреби користувачів та бізнесу при проектуванні архітектури.

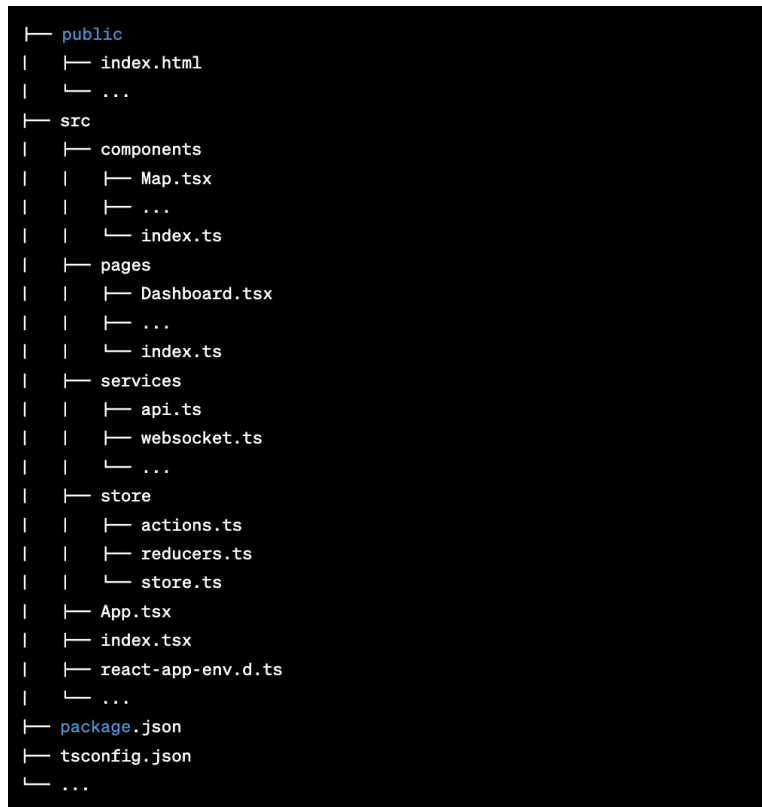


Рисунок 2.2 - Приклад структури проекту

Отже, створення архітектури розробки сайту є важливою дією, оскільки вона визначає успішність та ефективність сайту. Після побудови архітектури, можна одразу переходити до другої фази цього етапу, а саме до розробки дизайну, бо перед тим як нам писати функціонал, треба розробити дизайн усіх сторінок, щоб бачити як повинні виглядати наші елементи.

Наступний крок - розробка frontend та backend частин платформи. На цьому етапі ми розроблятиметься інтерфейс на основі дизайну та розробка функціоналу. Усі обрані бібліотеки встановимо в оточення, та використати враховуючи усі

вимоги. Також у ході розробки функціоналу, необхідно інтегрувати Mapbox в платформу та забезпечити відображення карт.

Останній етап - тестування нашої платформи. Після розробки веб-платформи важливо провести ретельне тестування для виявлення та виправлення можливих помилок, а також перевірки на відповідність вимогам. Для тестування можуть використовуватися різні підходи, такі як модульне тестування компонентів React.js, інтеграційне тестування взаємодії між компонентами.

Таким чином ми розглянули основні кроки розробки веб-додатку для логістики, використовуючи передові технології та бібліотеки. Розглянуто такі етапи як планування та аналіз вимог, проектування, розробка та тестування системи.

2.2 Вимоги до платформи розробки

Для того що б почати користуватися системою візуального бронювання нам необхідний будь-який встановлений браузер (наша система кросбраузерна, тому підійде будь-який браузер). Система повинна легко і швидко працювати на різних платформах.

Мінімальні системні вимоги для користування веб-додатком можуть змінюватися в залежності від конкретного додатку та його функцій. Однак, зазвичай, основні системні вимоги для користування веб-додатком включають:

- операційна система: Веб-додатки можуть працювати на будь-якій сучасній операційній системі, такі як Windows, macOS, Linux, Android або iOS;
- веб-браузер: Веб-додатки зазвичай працюють в будь-якому сучасному веб-браузері, такому як Google Chrome, Mozilla Firefox, Safari, або Microsoft Edge;
- швидкість Інтернет-з'єднання: Веб-додатки можуть працювати з будь-якої швидкості Інтернет-з'єднання, але, зазвичай, мінімальною швидкістю рекомендується 1 Мбіт/с.

Загальні системні вимоги для використання нашого додатку дуже низькі, і більшість комп'ютерів, що працюють на сучасних операційних системах, зможуть запуснути вебплатформу без проблем. Однак, є мінімальні вимоги для кращої продуктивності та користування функціями, які пропонує Google.

Нижче, у таблиці 2.1, зазначені мінімальні системні вимоги для використання нашого додатку на Windows, Mac та Linux.

Таблиця 2.1- Вимоги до різних операційних систем

Операційна система	Процесор	Вільне місце на диску	Оперативна пам'ять
Windows 7 або новіше	Intel Pentium 4 або вище, AMD Athlon 64 або вище	2ГБ	2ГБ
Mac OS X 10.10 або новіше	Intel-процесор	2ГБ	2ГБ
Linux (будь-який дистрибутив)	Intel Pentium 4 або вище, AMD Athlon 64 або вище	2ГБ	2ГБ

Важливо відзначити, що це лише мінімальні вимоги. Якщо є більше вільного місця на диску та більше оперативної пам'яті, буде змога працювати з більшим комфортом та швидкістю. Також, наявність більш потужного процесора може покращити швидкість завантаження сторінок та відповіді на запити.

Для розробки проекту використано середовище розробки Visual Studio Code. Цей редактор є одним з найпопулярніших інтегрованих середовищ розробки (IDE), що використовуються для розробки програмного забезпечення. Редактор можна встановити безкоштовно, що є ще однією перевагою над іншими платними аналогами, такими як: WebStorm, PhpStorm, IntelliJ IDEA, Komodo IDE, тощо. Але не дивлячись на те що редактор безкоштовний, це не робить його гіршим за платні, бо він має такі ж функції та навіть більше. VS Code підтримує понад 40 мов програмування, що означає, що можна використовувати VS Code для розробки програм у більшості популярних мов програмування. Також редактор має велику

кількість розширень, які можуть збільшити функціональність IDE. Ці розширення можна встановити безпосередньо з веб-магазину VS Code. Крім того VS Code має вбудовану підтримку Git, що дозволяє легко працювати з репозиторіями Git: можна створювати нові репозиторії, комітити зміни, відгалужувати та злити гілки, все з одного інтерфейсу. Ще одна перевага VS Code – це потужна навігаційна панель, яка дозволяє легко переходити між файлами та функціями. Інтерфейс програми зображено на рисунку 2.3.

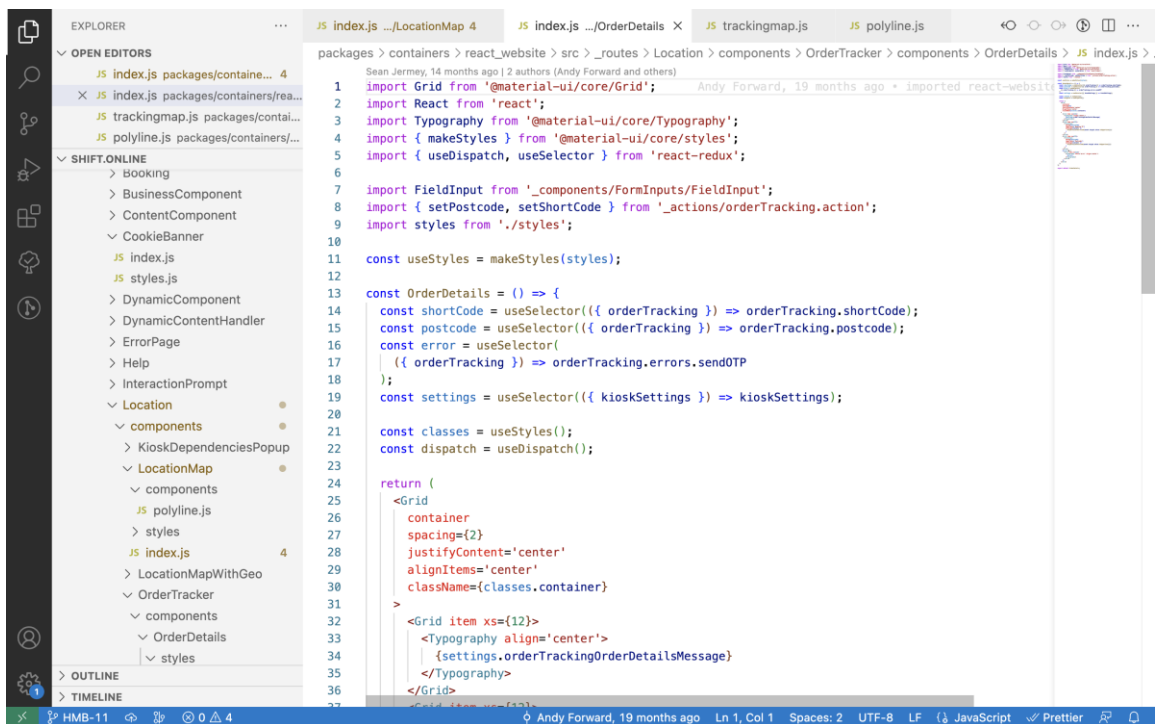


Рисунок 2.3 - Зображення інтерфейсу Visual Studio Code

Отже, ми визначились с вимогами до системи, та розібралися з особливостями редактору коду Visual Studio Code. Тож тепер ми можемо переходити до наступного кроку.

2.3 Аналіз вимог до функціональності та інтерфейсу транспортної логістичної вебплатформи

Основний етап нашої роботи - це визначення вимог до функціоналу транспортної вебплатформи. Отже, розпочнемо з малого. Наш додаток, свого роду адмін-панеллю, яка відстежує весь період існування замовлення, від створення до видалення. Так як в нас веб-сайт зв'язаний з логістикою, нам треба перевозити якийсь товар - це дерев'яні піддони. Тобто, ми можемо створювати замовлення на перевозку піддонів з точки А в точку Б. Але з усім по порядку. Спочатку треба розібратися з доступом користувачів до платформи, хто має право користуватись додатком, а хто ні.

У нашу систему не зможе зайти будь-який користувач, та подивитися контент, бо, як я казав вище, це адмін-панель, а отже доступ додаток має обмежений. На вході до платформи, нас зустрічає форма авторизації з двома полями: імейл, пароль. Якщо ми правильно введемо дані, то нас пустить до системи. Якщо ж ні, наприклад забули пароль, то в нас є можливість відправити на існуючу пошту імейл на відновлення пароля.

Додаток налічує 3 ролі користувачів: ADMIN, OPERATOR, USER. У кожного з ролей свій функціонал. Розпочнімо з можливостей USER. Можливості користувача дуже обмежені. Після того як USER увійшов до нашої системи, йому треба показати усі замовлення які створено на його ім'я, відображаються вони у вигляді таблиці. Користувач може створити замовлення, видалити його, та подивитися деталі після створення. Змінювати він його не може, бо це не логічно, адже може бути ситуація коли він захоче змінити якісь дані, а замовлення вже у дорозі їде до нього. У такому випадку треба створювати нове замовлення з новими вимогами. Для того щоб створити запит на доставку піддонів треба:

- заповнити ім'я отримувача, а також адресу куди доставлятиметься замовлення;
- заповнити дані о піддонах;

- заповнити обов’язкову дату відправки товару;

Після створення замовлення, користувачу треба зачекати поки ADMIN перевірить деталі запиту та підтвердить його або відхилити, також замовленню присвоюється статус “Очікує Схвалення” – To be approved. У нашій системі декілька статусів замовлення, послідовність присвоєння наведено на рисунку 2.4. Серед можливостей користувача є ще перегляд своїх даних в особистому кабінеті: імейл, номер телефону, тощо; можливість змінити пароль.

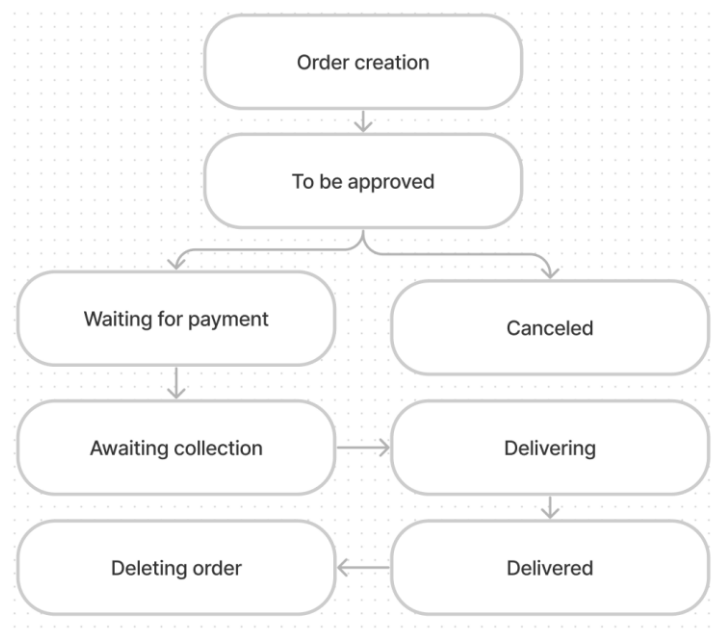


Рисунок 2.4 – Життєвий цикл статусів додатку

Тепер перейдемо до ролі ADMIN та OPERATOR. Їх можливості, однакові, єдина відмінність, що адмін може створювати, змінювати та видаляти операторів на відповідній сторінці, яка доступна тільки адміну. Ось до яких сторінок має доступ адмін та оператор:

- Deliveries – усі замовлення усіх користувачів у системі;
- Dashboard – статистика системи, доступна лише адмінам;
- Users – усі користувачі;
- Machines – мишини, на яких доставлятиметься товар;
- Unloading and loading – сторінка з розподіленням замовлень по машинам, та відображення шляхів перевозки на мапі;

- Personal profile – особистий кабінет, такий як у звичайного користувача;
- Operators – ця сторінка доступна лише адміну, тут можна створювати операторів та драйверів.

Як і звичайний користувач, адмін або оператор може створювати замовлення. Після створення, замовлення потрапляє на сторінку поставок, на якій є 3 вкладки у бічній панелі: підтверджені (approved), чекають підтвердження(to be approved), відхилені(canceled). Усі 3 сторінки в нас однотипні, з однаковим контентом: таблиця з замовленнями; модальне вікно з деталями; фільтри за допомогою яких є можливість сортувати контент сторінки. Коли замовлення потрапляє на сторінку очікування підтвердження, оператор може передивитися створений запит, внести зміни, щось додати або видалити, і потім або підтвердити або відхилити. Після цього, замовлення потрапляє на відповідні сторінки. Для того щоб замовлення відображалось на сторінці підтверджених товарів, потрібні дві обов'язкові речі: підтвердження оператором або адміном; і друге – це оплата товару. Після того як запит підтверджено оператором, у користувача в особистому кабінеті з'явиться відповідне повідомлення, яке наголошує про це. Функціонал надходження повідомлень розроблятиметься за допомогою WebSokcet. Користувачеві треба перейти на сторінку з деталями свого запиту, на якій, після підтвердження, з'явиться кнопка з посиланням на оплату товару. І вже після того як оплата зроблена – запит потрапить на сторінку підтверджених замовлень у оператора. Далі воно очікуватиме дати відправки товару, коли настане календарний день обраної дати – запит потрапить на сторінку Unloading and loading, де оператор зможе розподілити замовлення по машинах. У розподіленого замовлення, статус зміниться на «Delivering» – у цей період водій машина постачатиме замовлення. Маршрути замовлення з точки А до точки Б повинні відображатися на мапі особливим кольором відповідно до машини(у кожної машини свій колір відображення, щоб при відкритті мапи оператор зрозумі яка машина доставляє певний товар. Як замовлення дійде до адреси одержувача, статус зміниться на «Delivered». На мапі відображатиметься, що замовлення доставлене, а на

наступний день, видалиться із системи. Алгоритм роботи транспортної вебплатформи наведено у додатку (рис. A1).

Сторінка з машинами, операторами та персональні дані – усі ці сторінки однакові, тільки з різним контентом: таблиці з даними, фільтри, та можливість створення чи змінення відповідних даних через модальне вікно.

Dashboards – це остання сторінка операторської сторони, яку ми ще не розглянули. На даній сторінці є 2 графіки із статистикою. Перший – у вигляді кругової діаграми, на ньому зображено статистику відхилених, підтверджених та доставлених замовлень. Другий – гістограма, яка відображає популярність типів піддонів. Схема сторінок з доступами по ролям зазначено у додатку (рис. A2).

Важливим етапом нашої розробки є визначення вартості перевозки товару. Під час створення запиту, незалежно оператором чи користувачем, коли ми додаємо піддони, треба вказувати їх розміри: довжина, ширина, висота та вага. Виходячи з цього, наша система зрозуміє який тип треба привласнити піддону, це необхідно для подальшої оцінки піддону. Типи піддонів в залежать від їх висоти та ваги подані у таблиці 2.2.

Таблиця 2.2 – Типи піддонів

Тип	Висота	Вага
mini	не більше 60см	0-150кг
quarter	61-80см	0-300кг
half	81-120см	0-450кг
medium	121-160см	0-600кг
large	161-240см	451-750кг
full	0-240см	751-900кг
big	0-240см	901-1050кг
mega	0-240см	1050кг та вище

Стосовно ширини піддонів можуть бути: не менше 60х60, та не більше 300х120 або 240х160.

Після того як визначимо тип піддона, одразу треба розрахувати їх базову вартість. Окрім типів, для цього нам потрібна адреса доставки, а конкретніше область, куди доставлятиметься замовлення. Склад наших піддонів, звідки

відправлятимуться, знаходитиметься у Києві, а доставляти можна у будь яку область по Україні. Залежно від області та типу піддона, буде оголошена вартість пересилки за один піддон. Вартість пересилки залежно від областей наведено в таблиці Додатку А.

Окрім базової вартості, в додатку є ще фінальна вартість. Різниця в тому, що базова – це вартість піддону залежно від її розмірів, а фінальна – залежить від коефіцієнту надбавки за площу яку вони займатимуть. Піддон розміру «mini» не може оцінюватися, у разі переоцінки розмір піддону змінюється на «quarter» і до нього застосовуються доплати. Коли базова площа в м2 перевищує 1,28 м, вартість піддону необхідно переоцінити згідно з таблицею 2.2.

Таблиця 2.3 - Коефіцієнти надбавки залежно від площі

Область на основі м ²		Збільшити на %
Від	До	
0,00	1,28	0%
1,29	1,39	10%
1,40	1,59	30%
1,60	1,89	50%
1,90	2,27	80%
2,28	2,51	100%
2,52	2,87	130%
2,88	3,07	150%
3,08	3,35	180%
3,36	3,84	200%

Наприклад, в нас є 3 піддони розмірів 120x300см, висотою 70см, вагою 250кг та доставити треба у Запорізьку область. Висота та вага дає нам розуміння, що тип даних піддонів є «quarter», далі визначаємо базову вартість, площу, та кінцеву вартість за таким принципом:

$$\text{Базова вартість: } 1400 * 3 = 4200\text{грн}$$

$$\text{Площа: } 1,2 * 3 = 3,6\text{м}^2$$

$$\text{Кінцева вартість: } 4200 * (1 + 2(200\%)) = 12600\text{грн}$$

Розібралися з функціоналом, тепер перейдемо до вимог до інтерфейсу нашої вебплатформи. Вище ми трохи вже розібралися як виглядатимуть наші сторінки, в

основному це таблиці зі своїм контентом. Єдине що ми ще не знаємо, це який саме контент на кожній з них, тобто які стовпчики відображаються у таблицях, це проілюстровано на рисунку 2.5.

Ну і на останок, це шрифти з кольорами:

- шрифт - 'Raleway', sans-serif, базовий розмір 16px;
- колір тексту - #1B2936, колір кнопок - #FF640D;

Отриманий результат аналізу вимог до функціональності та інтерфейсу транспортної логістичної вебплатформи дозволяє зробити висновок, що для успішної реалізації проекту необхідно врахувати широкий спектр вимог, які пред'являються до системи. Виконання цих вимог допоможе створити високоефективну, безпечну та зручну веб-платформу.

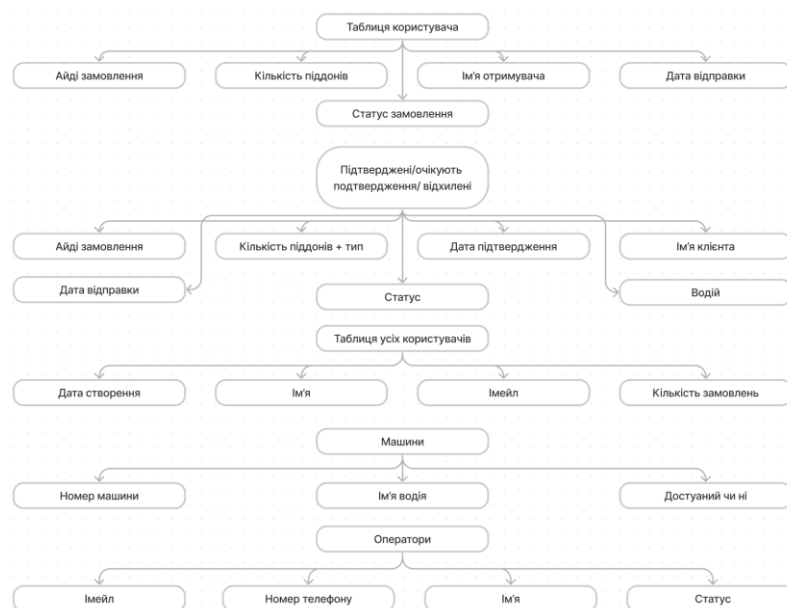


Рисунок 2.5 - Стовпчики усіх таблиць системи

2.4 Розробка архітектури системи

Ми визначились з вимогами до нашої вебплатформи, в нас залишився останній крок перед тим як розпочати розробку додатку – це спроектувати

архітектуру нашої системи. Розпочнемо з бекенду. Для створення нашої системи ми використовуватимемо Nest.js як фреймворк для створення серверної частини нашого додатку, а також PostgreSQL як базу даних для зберігання наших даних.

Nest.js - це фреймворк для створення масштабованих та ефективних серверних застосунків на Node.js. Він побудований на основі TypeScript і використовує принципи SOLID та DRY для розробки додатків з чистим та організованим кодом. Nest.js побудований на основі модульної структури, що дозволяє легко організовувати ваш код та повторно використовувати компоненти, також він підтримує TypeScript з коробки, що забезпечує зручний та безпечний розвиток додатків. Основна перевага Nest.js - вбудована підтримка WebSocket, що дозволяє створювати додатки в режимі реального часу. А це те що нам потрібно, адже ми повинні імплементувати функціонал повідомлень для користувачів та операторів як раз на WebSocket. Тож, Nest.js ідеальний фреймворк для нас. Тепер перейдемо до бази даних.

Існує безліч баз даних, різних за своїм призначенням, розміром, технологією та іншими характеристиками: MySQL, MongoDB, PostgreSQL, Microsoft SQL Server, SQLite, Redis, Cassandra, і тд. Ми використовуємо у розробці PostgreSQL. Чому саме цю бд? PostgreSQL - це потужна реляційна безкоштовна для використання база даних з відкритим вихідним кодом, яка використовує SQL (Structured Query Language) для керування даними. PostgreSQL розроблений для забезпечення високої рівної надійності, масштабованості і можливостей розширення, що робить його популярним рішенням для різноманітних застосувань. Порівняно з іншими реляційними базами даних, такими як MySQL та Microsoft SQL Server, PostgreSQL має декілька переваг. Наприклад, PostgreSQL має більш високий рівень стандартної сумісності зі стандартом SQL, а також більш широку підтримку розширень та засобів розробки. Крім того, PostgreSQL має більшу надійність та стійкість до відмов, що дозволяє забезпечувати високу доступність та продуктивність системи.

Розібралися з фреймворком та базою даних, тепер перейдемо до структури бекенду. Нижче на рисунку 2.5 наведена схема, яка показує, які компоненти включаються в нашу архітектуру.

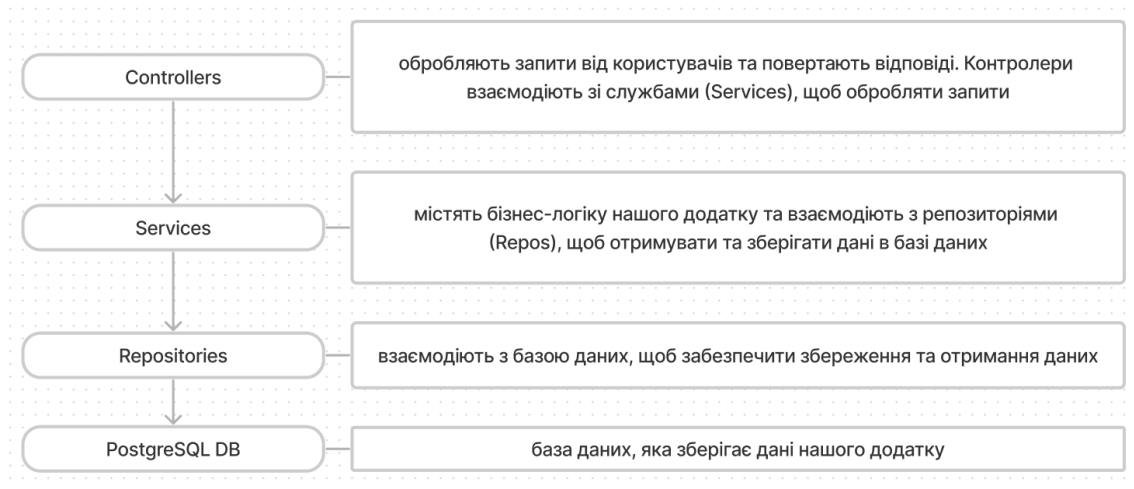


Рисунок 2.5 - Схема компонентів архітектури для Nest.js

Наступна послідовність використання показує, як наші компоненти взаємодіють один з одним для обробки запитів від користувачів:

- користувач робить запит до сервера;
- контролер отримує запит та передає його до сервісу;
- сервіс обробляє запит та взаємодіє з репозиторієм для забезпечення збереження або отримання даних з бази даних;
- репозиторій взаємодіє з PostgreSQL DB, щоб зберегти або отримати дані;
- репозиторій повертає дані сервісу, який повертає відповідь контролеру;
- контролер повертає відповідь користувачеві.

Тепер перейдемо до архітектури фронтенду. Як зазначено у першому розділі, для написання функціоналу на фронті, ми використовуємо React.js. але чому саме ця бібліотека, а не, до прикладу Vue.js чи Angular. Усі ці фреймворки дуже популярні зараз. Але все ж таки у React.js є свої переваги:

- простота та прозорість - React простіше вивчити та розуміти ніж Angular, а в порівнянні з Vue, React має більш прозору та логічну структуру;

- висока продуктивність - React працює швидко та ефективно завдяки використанню Virtual DOM. Це дозволяє мінімізувати кількість зайвих операцій з DOM, що поліпшує продуктивність додатків;

- більш гнучка архітектура - React не накладає обмежень на архітектуру проекту, що дозволяє розробникам вибирати та використовувати різні бібліотеки, які найкраще підходять для їхнього проекту;

- зручний для розробки SPA - React добре підходить для створення односторінкових додатків (SPA), тому що він дозволяє ефективно керувати станом додатка та забезпечує зручні інструменти для роботи зі сторінками.

Отже, дивлячись на усі ці переваги, можемо з упевненістю розробляти структуру нашого додатку на React. Вона складається з:

- `public` - це папка, де зберігаються статичні файли для веб-сайту, такі як HTML, зображення, шрифти і т.д;

- `src` - папка з вихідним кодом проекту. Вона містить компоненти, сторінки, сервіси, сховище (Redux), та інші файли, необхідні для роботи додатку;

- `assets` - папка де зберігатиметься картинки проекту, та основні стилі;

- `components` - папка з компонентами, які можуть бути використані на різних сторінках;

- `pages` - папка з компонентами, які представляють сторінку додатку;

- `services` - папка з файлами, які відповідають за зв'язок з API та WebSocket;

- `store` - папка з файлами, які відповідають за стан додатку за допомогою Redux;

- `index.tsx` - файл, який рендерить головний компонент в DOM;

- `react-app-env.d.ts` - файл, який містить типи, необхідні для роботи з React;

- `package.json` - файл залежностей та конфігурації проекту;

- `tsconfig.json` - файл конфігурації TypeScript;

- `.eslintrc.js` – файл конфігурації ESLint.

Ми створили структуру нашого проекту. Тепер треба налаштувати маршрутизацію нашої платформи. Це перший крок нашої розробки. Як зазначено у вимогах, в нас 3 ролі користувачів, у кожного з яких є доступ до певних сторінок. Отже, нам треба написати функціонал, який забороняє або дозволяє доступ до певних сторінок. Для цього нам потрібно створити масив усіх роутів, з параметром «roles», в якому зазначено перелік ролей, які мають доступ до сторінки, на лістингу 2.1 наведена його реалізація.

Лістинг 2.1 – Масив роутів вебплатформи

```
export const routes = [{
  path: '/deliveries/approved',
  exact: true,
  role: [RoleEnums.OPERATOR, RoleEnums.ADMIN],
  element: (
    <Deliveries>
      <ShippingApprovedContainer/>
    </Deliveries>
  ),
},
{
  path: '/operators_list',
  exact: true,
  role: [RoleEnums.ADMIN],
  element: <OperatorsList/>,
},
{
  path: '/shipments_list',
  exact: true,
  role: [RoleEnums.USER],
  element: <PrivateAreaShippingList/>,
}
];
```

Але цього нам замало, бо це просто масив, перелік усіх сторінок системи які налічує наш додаток. Для того щоб прогнати цей масив через цикл та послатися на параметр roles, нам потрібно спочатку зрозуміти під якою роллю ми авторизувались. Це зробити дуже просто. Після того як користувач увійде у систему, йому наша API видасть токен авторизації, і сама завдяки цьому токenu, система зрозуміє роль юзера: адмін, оператор чи звичайний користувач. Для чого взагалі потрібен токен? Токен гарантує безпеку доступу до конфіденційної інформації та ресурсів від повідомлень до баз даних. Тобто деякі запити до нашої

API обмежені цим токеном, це дасть нам упевненість що тільки користувачі нашої системи зможуть мати доступ до наших даних. Також, токени можуть бути обмежені в часі дії і обмеженнях доступу, що дозволяє контролювати доступ до конкретних функцій чи інформації. Отже, після успішної авторизації, API видасть нам токен, яких ми збережемо у localStorage - механізм зберігання даних в браузері, який дозволяє зберігати дані в локальному сховищі на комп'ютері користувача. Тепер ми можемо прогнати через цикл наш масив з роутами. Перелік зазначено у лістингу 2.2

Лістинг 2.2 – Конфігурація маршрутизатора на основі масиву роутів та ролі користувача

```
import { Routes, Route, matchRoutes, useLocation, Navigate } from
'react-router-dom';
import { useAppSelector } from 'store/store';
import { RoleEnums } from 'enums';
import { routes } from './index';
const ALL_ADMIN_ROLES = [RoleEnums.OPERATOR, RoleEnums.ADMIN];
const CustomRoute = () => {
  const role = useAppSelector((state) =>
state.auth.tokenStructure?.role);
  const location = useLocation();
  const match = matchRoutes(routes, location);
  const route = match && match[0].route;
  if ((!role && route?.role.length) || (role && route?.role.length
&& !route.role.includes(role))) { return <Navigate to="/" />; }

  if (!route?.role.length && role && ALL_ADMIN_ROLES.includes(role))
  {
    return <Navigate to="/deliveries" />;
  }
  if (!route?.role.length && role === RoleEnums.USER) {
    return <Navigate to="/shipments_list" />;
  }
  if (route?.element) {
    return (
      <Routes>
        <Route path={route.path} element={route?.element} />
      </Routes>
    );
  }
  return null;
};
export default CustomRoute;
```

2.5 Висновки до розділу 2

У цьому розділі було розглянуто основні кроки розробки веб-додатку для логістики, використовуючи передові технології та бібліотеки. Розглянуто такі етапи як планування та аналіз вимог, проектування, розробка та тестування системи.

Також, визначились с вимогами до системи, та розібралися з особливостями редактору коду Visual Studio Code. Отриманий результат аналізу вимог до функціональності та інтерфейсу транспортної логістичної вебплатформи дозволяє зробити висновок, що для успішної реалізації проекту необхідно врахувати широкий спектр вимог, які пред'являються до системи. Виконання цих вимог допоможе створити високоефективну, безпечну та зручну веб-платформу.

Ми обрали базу даних, розібралися як у нас працює взаємозв'язок між користувачем та API. Також побудували структуру та маршрутизатор. Використання обраних бібліотек дозволяє створювати високоякісну вебплатформу з високою продуктивністю та швидкістю роботи. Ці технології допомагають зменшити час розробки, спростити тестування та підтримку, а також підвищити рівень безпеки та стабільності роботи системи.

3 РЕАЛІЗАЦІЯ, ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

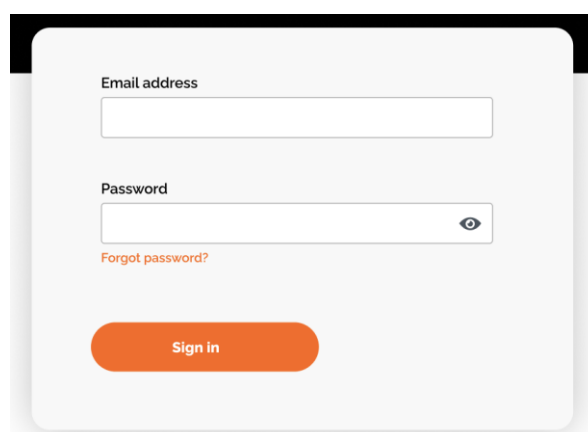
3.1 Розробка дизайну за допомогою онлайн-інструменту Figma

Перш ніж розпочати розробляти функціонал нашої системи, треба створити дизайн по вимогам, які прописані у другому розділі. Як вже зазначалось, ми використовуватимемо онлайн-інструмент Figma.

Отже, для того щоб розпочати робити дизайн, нам треба перейти за посиланням, авторизуватись і створити новий файл. Далі, у цьому файлі, треба зробити три секції, у яких будемо розробляти дизайн наших сторінок:

- auth section: секція, у якій будуть сторінки авторизації та оновлення паролю;
- user section: тут будуть усі сторінки для ролі USER;
- admin section: сторінки для ролі ADMIN та OPERATOR.

Далі, створюємо рамки, всередині яких, якраз реалізовано дизайн наших сторінок. Розпочнемо з форми авторизації. Спочатку робимо header нашої сторінки. Так як форма авторизації доступна лише користувачам, котрі ще не пройшли верифікацію, тобто ніяких доступів до сторінок нашої системи вони не можуть мати, а отже на цій сторінці у нас буде тільки лого нашого додатку. Після авторизації, до header додається посилання до певних сторінок. Далі, я створив заголовок, який дає користувачеві зрозуміти що це за форум. Ну і на останок, сама форма авторизації з двома полями email та password: підзаголовки над текстовими полями, самі текстові поля, посилання на форму відновлення паролю, та кнопка відправки пароля на нашу API для перевірки даних. Нижче на рисунку 3.1 проілюстровано реалізація форми нашої системи.



The image shows a login form with a light gray background and rounded corners. It contains the following elements:

- A label "Email address" above a white text input field.
- A label "Password" above a white text input field with a small eye icon on the right for toggling visibility.
- A link "Forgot password?" in orange text below the password field.
- An orange rounded button with the text "Sign in" in white, centered at the bottom.

Рисунок 3.1 - Форма авторизації транспортної вебплатформи

Звичайно усі шрифти та кольори були задіяні за вимогами. Доказ цього зображений на рисунку 3.2

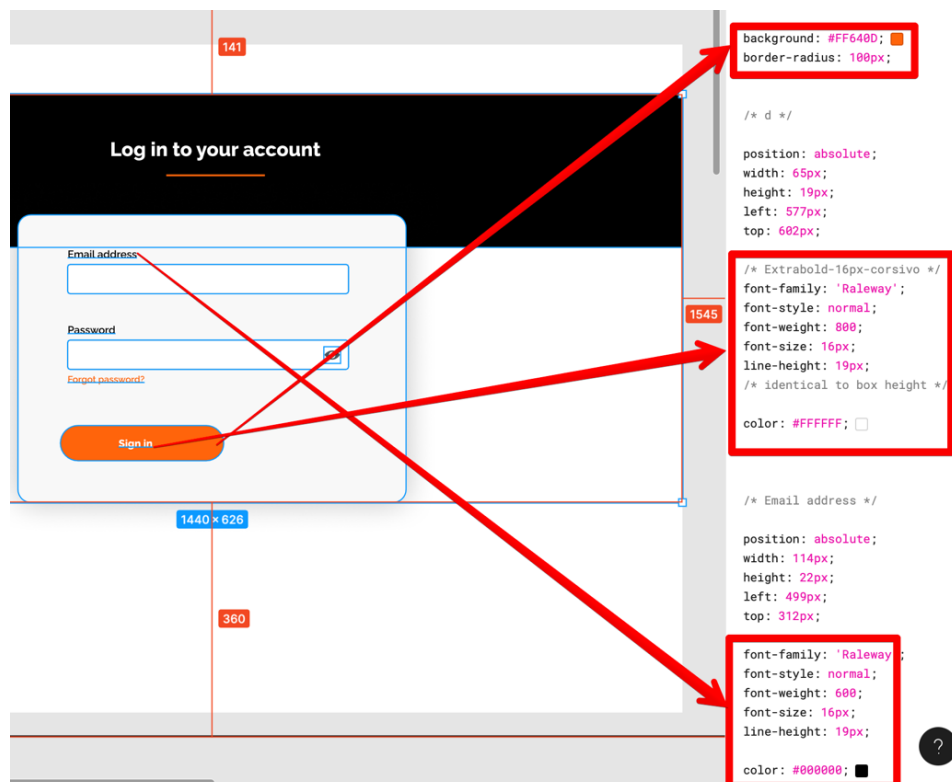


Рисунок 3.2 - Застосування стилів до дизайну

Як ми бачимо, усі шрифти, кольори та концепція застосовані за вимогами. Подальша реалізація дизайну інших сторінок буде проходити таким же чином, , отже нема сенсу описувати кожен сторінку. Реалізація інших сторінок проілюстровано у додатку (рис. А3).

3.2 Розробка функціоналу за допомогою React.js та Nest.js

Ми створили дизайн нашої платформи, у нас є усі сторінки, тепер нам треба зробити верстку та написати функціонал системи. Знову ж таки, розпочнемо з авторизації, бо, як зазначалось вище, щоб взаємодіяти з нашою системою, нам потрібен токен авторизації. Як це буде працювати з функціональної сторони? Спочатку створюємо файл конфігурації для наших запитів зі сторони фронт-енду, реалізація написана у лістингу 3.1.

Лістинг 3.1 - Конфігурація запитів на фронт-енді

```
export const instance = axios.create({
  baseURL: `${AppConfig.API_URL}/api`,
  headers: { 'Content-Type': 'application/json' },
});
instance.interceptors.request.use(
  async (config) => {
    if (config.headers) {
      const token = localStorage.getItem('token');
      if (token) {
        config.headers.Authorization = `Bearer ${token}`;
      }
    }
    return config;
  }, (error) => Promise.reject(error),
);
export const SignIn = (data) => instance.post('/auth/login', data);
```

У цьому коді я створив екземпляр Axios за допомогою `axios.create()`. Він приймає об'єкт конфігурації, де вказується базовий URL для всіх запитів (`baseURL`) та заголовки запиту (`headers`). Заголовок вказує серверу, що дані, які надсилаються, мають формат JSON. Також важливо прописати перехопник запиту (`request interceptor`) за допомогою методу `interceptors.request.use()`. Цей перехопник викликається перед кожним відправленням запитом і дозволяє модифікувати його конфігурацію. Всередині прописана функція, яка перевіряє наявність токена у локальному сховищі, якщо токен існує, встановлюється заголовок `Authorization` зі значенням `"Bearer ${token}"`. Саме завдяки цьому наш сервер зможе перевірити автентифікацію клієнта. Ну і в самому кінці прописана функція `SignIn`, це і є наш запит за шляхом `"/auth/login"` та тілом, в якому дані для передачі на сервер.

Отже ми написали наш перший запит, але нам треба його викликати. З цим нам допоможе `Redux`. Заповнюємо усі поля форми авторизації і викликаємо `action creator`, в який передаємо введені дані, всередині цієї функції робиться запит на сервер. У тілі запиту відправляємо два поля у форматі `"{ email: 'email@test.com', password: 'testpassword' }"`. Далі, на стороні бекенду, перевіряється наявність цих даних у системі. На лістингу 3.2 показана реалізація обробка запиту сервером.

Лістинг 3.2 - Обробка запиту авторизації.

```
@ApiTags('auth')
@Controller('auth')
export class AuthController {
  constructor(
    private readonly authService: AuthService,
  ) {}
  @ApiResponse({ status: 200, type: LoginOutputDto })
  @ApiResponse({
    status: 422,
    description: 'Unprocessable entity',
  })
  @ApiBody({ type: LoginUserDto })
  @UseGuards(LocalAuthGuard)
  @Post('login')
  async login(@Request() req) {
    const login = await this.authService.login(req.user);
    return login;
  }
}
```

Розглянемо детально, що тут написано:

- `@ApiTags('auth')` визначає тег або мітку для цього контролера, що допомагає в організації документації API;
- `@Controller('auth')` визначає маршрут, за яким будуть доступні всі ендпоінти цього контролера. У цьому випадку, всі ендпоінти будуть доступні за шляхом `“/auth”`;
- конструктор `constructor(private readonly authService: AuthService)` використовується для впровадження класу `AuthService` в контролер. Це дозволяє використовувати функціонал `AuthService` в методах контролера;
- `@ApiResponse({ status: 200, type: LoginOutputDto })` визначає відповідь, яку поверне цей ендпоінт в разі успішного виконання. У цьому випадку, статус відповіді буде 200, а типом відповіді буде `LoginOutputDto`;
- `@ApiResponse({ status: 422, description: 'Unprocessable entity' })` визначає відповідь, яку поверне цей ендпоінт в разі, коли запит містить невалідні або неприпустимі дані. У цьому випадку, статус відповіді буде 422, а описом помилки буде `“Unprocessable entity”`;

- `@UseGuards(LocalAuthGuard)` визначає, що цей ендпоінт буде захищений за допомогою стратегії аутентифікації `LocalAuthGuard`. Це означає, що перед виконанням цього методу буде проведена аутентифікація користувача. Реалізація стратегії наведена у лістингу 3.3;
- `@Post('login')` визначає, що цей ендпоінт буде доступний тільки за HTTP методом POST і за шляхом `"/auth/login"`;
- `async login(@Request() req)` визначає функцію обробки запиту для цього ендпоінту. В цьому випадку, це функція `login`, яка очікує об'єкт `req` в якості параметру. `@Request()` вказує на те, що параметр `req` має бути витягнутий з об'єкту запиту, який надається фреймворком `Nest.js`.

Лістинг 3.3 - Перевірка даних користувача.

```
async validateUser(email: string, password: string) {
  const user = await this.userService.findByEmail(email);
  if (!user) return throw new UnauthorizedException();
  const checker = await compare(password, user.password);
  if (!checker) return throw new UnauthorizedException();
  return user;
}
```

У тілі методу, спочатку викликається функція `login` класу `authService`, передаючи `req.user` як аргумент. Всередині цієї функції створюється токен авторизації, функціонал наведено у лістингу 3.4. Оскільки `req` є об'єктом запиту, `req.user` відноситься до користувача, який встановлений під час аутентифікації. Результат функції `login` повертається як відповідь на запит.

Лістинг 3.4 - Створення токена авторизації

```
async login(user: User) {
  const payload = {
    id: user.id,
    email: user.email,
    role: user.role,
  };
  const accessToken = this.jwtService.sign(
    payload,
    this.tokenOptions('access'),
  );
  const refreshToken = this.jwtService.sign(
```

```

    { ...payload, type: 'refresh' },
    this.tokenOptions('refresh'),
  );
  return { isAuthorized: true, accessToken, refreshToken };
}

```

Отже, клас `AuthController` обробляє POST-запит за шляхом `"/auth/login"`, приймає об'єкт `LoginUserDto` в тілі запиту, виконує аутентифікацію залежністю `LocalAuthGuard` і повертає результат функції `login` класу `authService` як відповідь.

При успішному запиті, на фронті ми викликаємо функцію-ред'юсер куди передаємо відповідь, тобто токен, і вже у ред'юсері ми зберігаємо цей токен у `localStorage`, і все, ми пройшли авторизацію, тепер можемо користуватися нашим додатком.

Для того щоб створити перше замовлення, спочатку треба додати у систему оператора та користувача. Розпочнемо з оператора. Для цього нам треба придумати ім'я, ім'я, номер телефона, присвоїти роль(оператор чи водій) та згенерувати пароль. На рисунку 3.3 проілюстрована заповнена форма.

Рисунок 3.3 - Форма створення оператора

Для того щоб кожного разу не вигадувати новий пароль, у нашій платформі реалізовано генерація випадкового пароля. По-перше це спростить створення, бо кожного разу вигадувати новий пароль не вистачить фантазії. По-друге, пароль

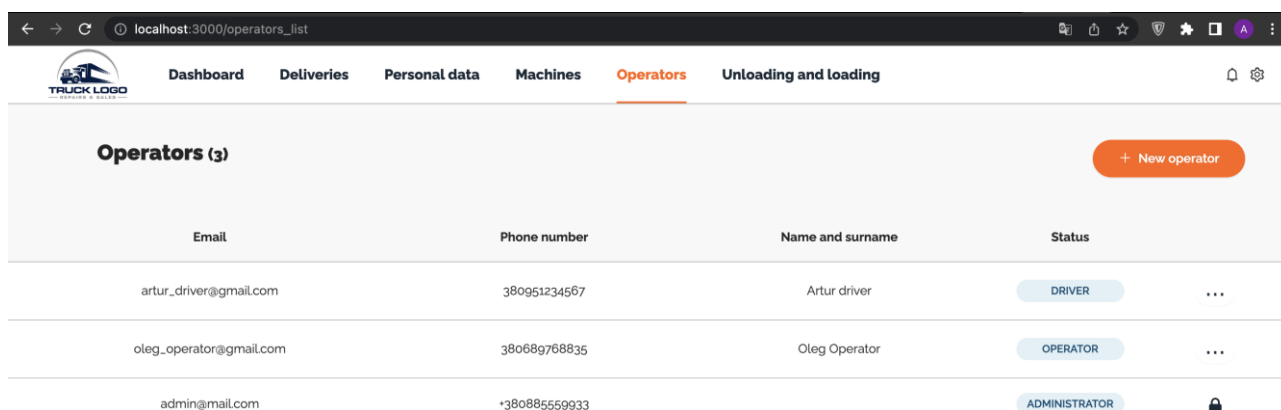
повинен бути надійним, налічувати хоча б один спеціальний символ, одну заголовну букву, та довжиною мінімум 8 символів, наш генератор як раз створює пароль по таким вимогам. Потім, коли ми відправляємо на сервер дані, ми повинні захешувати наш пароль через хеш-функцію. Основна особливість хеш-функцій полягає в тому, що вони є односторонніми. Це означає, що з хеш-коду неможливо одержати вихідний пароль. При перевірці пароля, введенного користувачем, система хешує введений пароль і порівнює його хеш-код з збереженим хеш-кодом у базі даних. Якщо хеш-коди співпадають, пароль вважається правильним.

Реалізація створення оператора та збереження у базу даних реалізовано у лістингу 3.5.

Лістинг 3.5 - Створення оператора та збереження у базу даних

```
async createUser(dto: CreateUserDto): Promise<User> {
  let data = dto;
  data = {
    ...dto,
    password: hashSync(dto.password),
  };
  const user = await this.userSchema.create(data);
  return user;
}
```

Після створення можемо переглянути перелік усіх наших операторів, адмінів та водіїв. Я одразу створив водія, щоб він також вже був у нашій системі. Таблиця з операторами проілюстровано у рисунку 3.4.



Email	Phone number	Name and surname	Status
artur_driver@gmail.com	380951234567	Artur driver	DRIVER
oleg_operator@gmail.com	380689768835	Oleg Operator	OPERATOR
admin@mail.com	+380885559933		ADMINISTRATOR

Рисунок 3.4 - Таблиця операторів, адмінів, та драйверів

Створення користувача відбувається таким же чином, тож цю частину можемо пропустити. Я створив користувача на ім'я Alexey Balabuha, та імейлом balabuha.a.ogl@gmail.com, тож саме на цього користувача створюватимемо усі замовлення системи.

Створимо наше перше замовлення. Спочатку треба обрати користувача, на якого буде оформлене це замовлення. Зробити ми це можемо за допомогою <Select> компоненту бібліотеки ant-design, у цьому селекті буде перелік імейлів усіх наявних користувачів системи. На рисунку 3.5 зображено як виглядає цей селект.

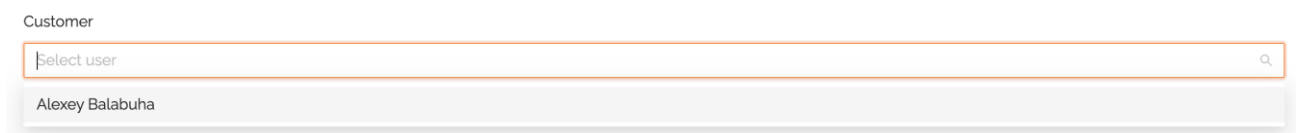


Рисунок 3.5 – Селект вибору користувача

Далі, треба заповнити усі поля, зображені на рисунку 3.5, я зробив валідацію усіх полей, щоб дати зрозуміти користувачеві, що треба заповнити перед відправкою даних.

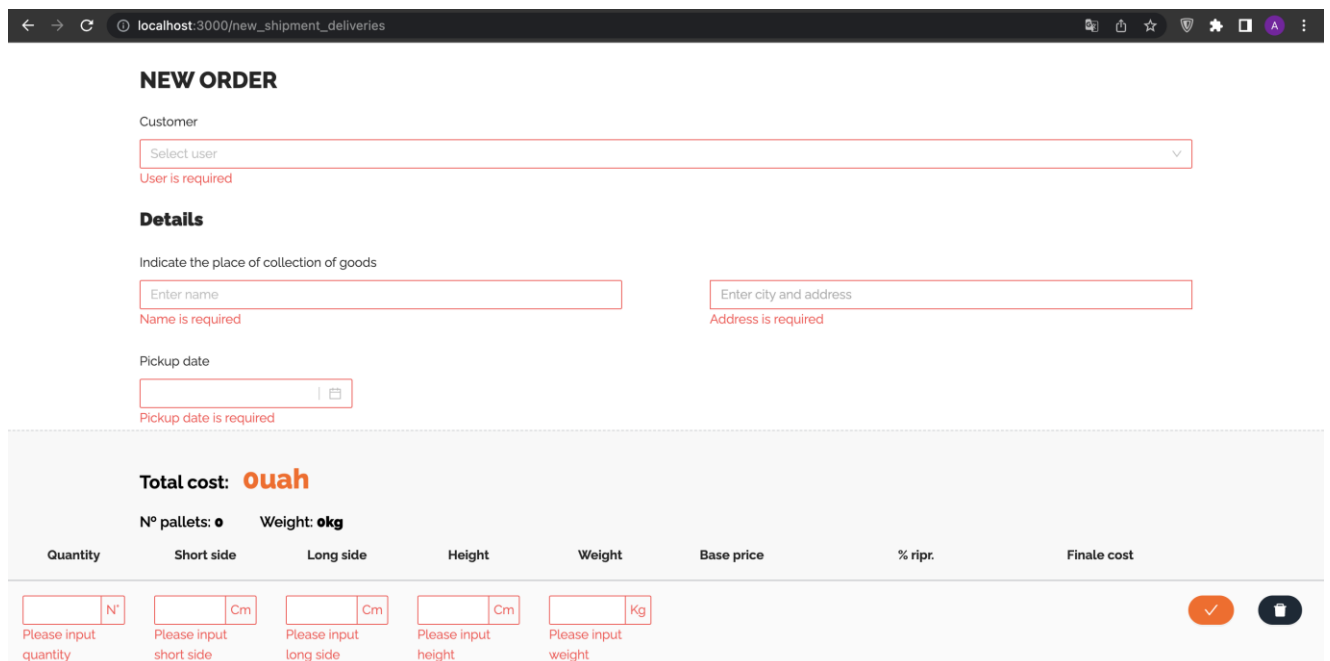


Рисунок 3.5 – Поля, які заповнюються для створення замовлення

Перейдемо до основної та одної з найскладніших частин нашого функціоналу – це прорахунок вартості піддонів. Як зазначено у другому розділі, нам для цього потрібно дані про піддони та область доставки. Нагадаємо, що відправка піддонів буде виконуватись із Київської області. Після відправки даних, на стороні бекенду визначиться коефіцієнт надбавки до ціни, сімейство піддонів, базова вартість та кінцева вартість. Почнемо з визначення коефіцієнту та сімейства. У лістингу 3.6 написана функція, яка визначає ці дані.

Лістинг 3.6 – Визначення коефіцієнту та сімейства піддонів

```
async calculatePalletPercentageAndFamily(pallet:
PalletInterface): Promise<PalletEstimationDto> {
  const coef: number = (pallet.depth / 100) * (pallet.width /
100);
  const family: DimensionFamily =
    await this.dimensionEstimationRepository.findFamily(
      pallet.height,
      pallet.weight,
    );
  const coefficientPercent: number =
    await this.coefficientEstimationRepository.findPercent(coef);
  const percent = coefficientPercent / 100;
  return { percent, family };
}
```

За допомогою методу `findFamily` класу `DimensionEstimationRepository` відбувається з'єднання з базою даних, та з відти дістається потрібний нам тип. Дані, які прописані у другому розділі(дивись таблицю 2.2 та таблицю 2.3), заздалегідь додано у базу даних, тож тепер ми можемо просто передавати потрібні нам дані, та з бази будуть підтягуватись потрібні нам параметри. Метод `findPercent` працює аналогічно з `findFamily`. Потім, ми прораховуємо вартість нашої доставки, прорахунок зображений на лістингу 3.7.

Лістинг 3.7 – Прорахунок вартості замовлення

```
public async getPalletPriceWithCoef(
  address: AddressInterface,
  percentAndFamily: DimensionFamily,
  pallet: PalletInterface,
): Promise<{ price: number; base_price: number; coef: string }> {
```

```

    const basePrice = await this.searchPrice(address,
percentAndFamily.family);
    const reCalcPrice = basePrice * (1 + percentAndFamily.percent);
    return {
      price: +reCalcPrice.toFixed(2),
      base_price: +basePrice.toFixed(2),
      coef: ` ${+percentAndFamily.percent.toFixed(2) * 100}%`,
    };
  }
}

```

Метод класу `searchPrice` приймає два параметри, це адреса та тип піддону. Ці параметри нам потрібні для того щоб також звернутися до нашої бази даних та дістати з відти вартість залежно від області доставки та типу піддону(дивись додаток А таблиця А.1). Ну і останнє що в нас залишається – це помножити вартість на кількість піддонів, і все, наша вартість прорахована. На рисунку 3.6 зображено як виглядають додані піддони та їх прорахована вартість. Для місця доставки обрана Запорізька область.

Total cost: 23820 uah							
N° pallets: 7		Weight: 350kg					
Quantity	Short side	Long side	Height	Weight	Base price	% ripr.	Finale cost
6	250	100	70	300	8400	130%	19320 ...
1	150	200	90	50	1800	150%	4500 ...

Рисунок 3.6 – Введені дані піддонів та їх прорахована вартість

Чудово, в нас створено наше перше замовлення. Форма створення замовлення прописана у додатку Б.1. Воно потрапляє до таблиці “To be approved” зі статусом “Waiting for payment”. Тепер користувач, якого було обрано в селекті, повинен оплатити замовлення. Отже, ми заходимо під обліковими даними цього клієнта. По дизайну, в нас у верхньому правому куту є дзвіночок, який відповідає за повідомлення для користувача, що на нього було створено замовлення. Надходження цих повідомлень реалізовано на WebSocket. Функціонал WebSocket зазначено на лістингу 3.8.

Лістинг 3.8 – Функціонал WebSocket.

```
useEffect(() => {
  socket.connect();
  socket.on('notification', (data) =>
dispatch(setNotifications(data.message.reverse())));
  return () => {
    socket.off('notification');
    dispatch(clearNotifications());
  };
}, [socket, dispatch]);
useEffect(() => {
  socket.on('new_notification', (data) =>
dispatch(setNewNotifications(data.message)));
  return () => {socket.off('new_notification')};
}, [socket, notReadNotifications, dispatch]);
useEffect(() => {
  socket.on('success', (data) =>
dispatch(readNotification(data.message[0])));
  return () => { socket.off('success') };
});
const emitNotificationHandler = (id) => {
  socket.emit('acknowledgement', { notification_id: id });
};
```

Розберемо цей код:

- `socket.connect()` - викликається для підключення до сокетного сервера;
- подія “notification” – встановлюється прослуховування події ‘notification’ з сокетного сервера. Коли отримується подія, виконується callback-функція, яка отримує дані data. Далі, ця data зберігається у redux за допомогою action creator `setNotifications`;

- подія “new_notification” - вона викликається коли надходить нове повідомлення;

- подія “acknowledgement” – ця подія викликається після того, як користувач натискає на повідомлення. Ця подія потрібна для того щоб прочитати це повідомлення, щоб дати зрозуміти користувачеві, що він вже взаємодіяв з ним;

- подія “success” – вона викликається одразу після “acknowledgement”, щоб при успішному прочитанні, оновити наше сховище redux.

На рисунку 3.7 зображено як виглядають прочитані та не прочитані повідомлення у нашій системі.

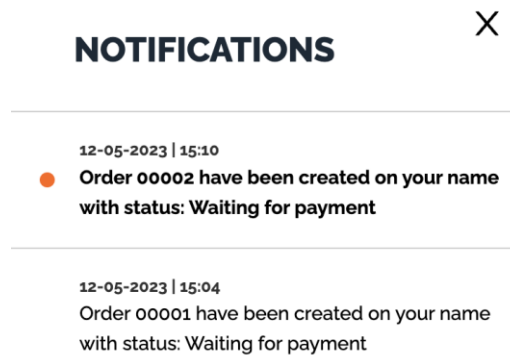


Рисунок 3.7 – Вигляд повідомлень транспортної вебплатформи

Тепер перейдемо до оплати нашого запиту. Оплата здійснюється за допомогою stripe. Stripe є однією з найпопулярніших платіжних систем, яка надає можливість приймати онлайн-платежі через додатки і веб-сайти. Ця платформа пропонує простий API та зрозумілу документацію, що робить його легким у використанні навіть для початківців. Stripe підтримує різні типи платежів, включаючи кредитні та дебетові картки, електронні гаманці (наприклад, Apple Pay та Google Pay) та банківські перекази. Коли користувач натисне на кнопку оплатити, зробиться запит на нашу API, який повинен повернути строку згенерованого посилання на наш stripe. Після того як користувач зробить оплату, його переносить на сторінку з усіма його замовленнями. На лістингу 3.9 написаний функціонал генерації посилання яке веде на stripe.

Лістинг 3.9 – Генерація посилання на stripe.

```
const _price = await this.stripeClient.prices.create({
  unit_amount:
    Math.trunc(+order.price.toFixed(2) * (100 + this.TAX)) +
    this.getStripeFee(+order.price),
  currency: order.currency,
  product: _product.id,
});
const paymentLink = await
this.stripeClient.paymentLinks.create({
  line_items: [{ price: _price.id, quantity: 1 }],
  after_completion: {
    type: 'redirect',
    redirect: {
      url: `${config.stripe.successURL}/shipments_list`,
    },
  },
});
```

Після оплати, наше замовлення отримує статус “Awaiting collection”. Тепер замовлення буде очікувати дати відправки, а як дочекається, потрапить до секції “Unloading and loading”, де оператор зможе розподілити їх по машинам. Коли замовлення розподілять, статус змінюється на “Delivering”, а як дійде до місця призначення – “Delivered”. Змінювати статус замовлення, оператор може у самій таблиці, у стовпчику “Status”, там відображається статус замовлення у вигляді селекту, де буде перелік усіх доступних статусів, які можна присвоїти замовленню. На рисунку 3.8 зображено таблицю підтверджених замовлень, на якому також показані ці селекти.

Id	N°pallet ↕	Approved date ↕	Client name ↕	Pickup date ↕	Status	Driver ↕
0002	1 - HALF	14-05-2023	Alexey Balabuha	14-05-2023	Awaiting collection ▾	- ...
00001	6 - QUARTER 1 - HALF	14-05-2023	Alexey Balabuha	14-05-2023	Awaiting collection Delivering Delivered	- ...

< 1 > 10 / pagina ▾

Рисунок 3.8 – Таблиця підтверджених замовлень

Як показано на рисунку, в нас обрана дата 14-те травня, отже, воно повинно потрапити до секції “Unloading and loading” з цієї дати, або більшої. Але, спочатку треба створити машину, на якій ми будемо доставляти товар. Нижче на рисунку 3.9 проілюстровані машини, на які створені у нашій системі.

Machines (2)			+ New machine
License plate	Drivers name	Availability	
AA7755AA	Artur driver	✓	...
AA3423AA		✓	...

Рисунок 3.9 – Таблиця машин

Як можна помітити, у другій машини нема водія. Це тому що у нашій системі створено лише одного водія, і неправильно було б на його додати до двох машин.

Тож, якщо ми захочемо додати замовлення на цю машину, треба створити нового драйвера, та прив'язати його до цієї машини.

В нас є два замовлення, водій з машиною, потрібна дата для розподілення, тож тепер ми можемо перейти до секції, де ми розподілятимемо наші замовлення по машинам і відображатимемо шляхи на мапі. Як це працює? Коли ми заходимо на сторінку “Unloading and loading”, в нас відображаються стовпчики з замовленнями(дивись рисунок A3). Перший стовпчик – це замовлення, які ще не розподілені по системі. Інші – це машини, наявні у нашій платформі, на які як раз можна їх розподілити. Щоб розподіляти запити реалізовано функціонал drag-n-drop(перетягни та впусти), тобто, нам треба перетягнути замовлення зі стовпчика нерозподілених замовлень, на нашу машину. Допомогти реалізувати цей функціонал мені допомогла спеціальна бібліотека, яка називається “react-dnd”. Функціонал зазначено у лістингу 3.10.

Лістинг 3.10 – Drag-n-drop функціонал

```
const [, drag] = useDrag(
  () => ({
    type: ItemTypes.CARD,
    canDrag: !completedTrip,
    item: { shippingRequest },
  }),
  [shippingRequest.id],
);
const [, drop] = useDrop(
  () => ({
    accept: ItemTypes.CARD,
    drop({ shippingRequest }: DropType) {
      moveCardToColumn(shippingRequest);
    },
  }), [moveCardToColumn],
);
```

Розберемо код:

- “drag” – перетягує наш компонент;
- “drop” – відповідно, впускає у області, куди ми перетягли;
- параметри “type” та “асерт” – дають зрозуміти бібліотеці, що ці дві змінні між собою зв’язані;

- “sandrop” – забороняє або дозволяє перетягувати елемент. Це нам потрібно для того, щоб заборотини перетягувати замовлення, коли воно вже доставлено, адже ми маємо можливість між машинами також перетягувати замовлення;
- “item” – об’єкт, куди ми передаємо об’єкт замовлення, що перетягується;
- “drop” – викликається після того як замовлення перетянулось, всередині цього метода ми визиваємо функцію, яка оновляє наше сховище та робить запит на сервер.

Одразу після того як ми присвоїли замовлення на машину, у нас оновлюється мапа. На ній з’являється крапка, куди повинно доставитись замовлення та промальовується шлях. Також, як зазначено у дизайні, додав блок, де можна чекбоксами вмикати чи вимикати відображення шляхів машин. Реалізація функціоналу мапи подана у додатку В.

У таблиці підтверджених замовлень у колонці статусів, в одного із замовлень я обрав що воно доставлене, для прикладу. Нижче на рисунку 3.10 показано відображення наших замовлень на мапі.

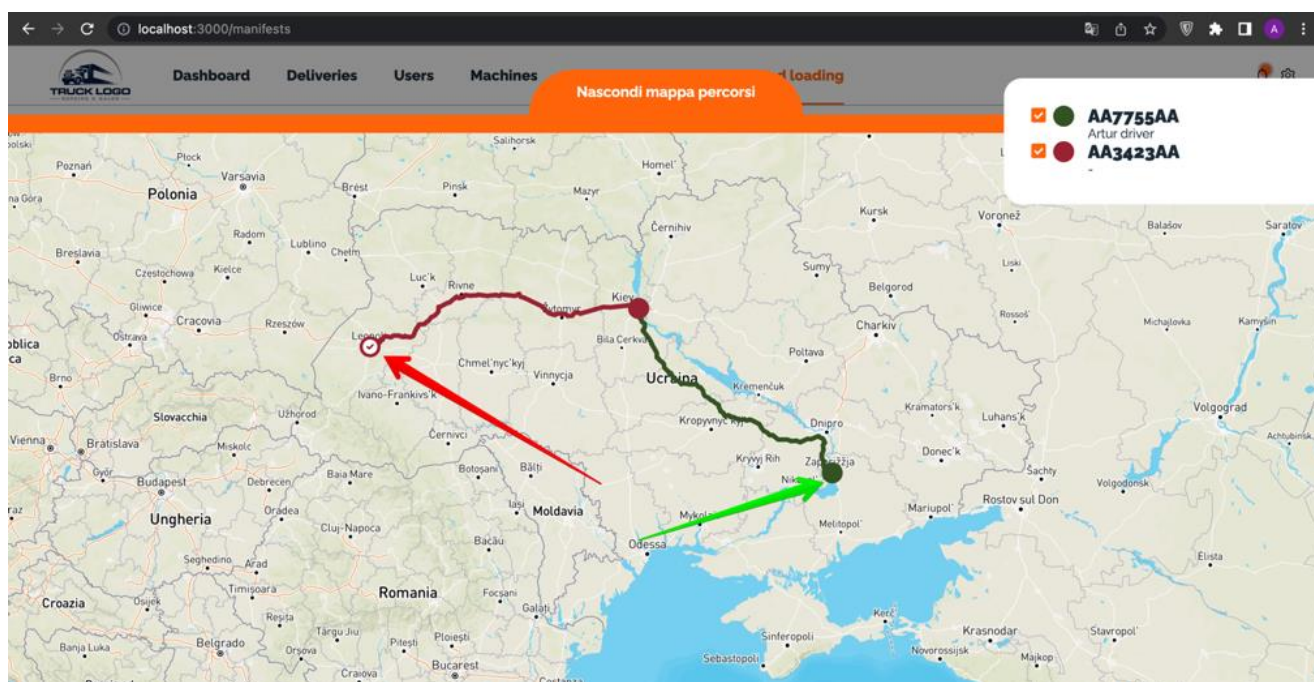


Рисунок 3.10 – Шляхи доставок на мапі

Як ми бачимо, доставка що прямувала до Львова завершена, і на ній відображається відповідна іконка, а до Запоріжжя ще доставляється.

Отже, ми майже все зробили від створення користувачів, до відображення мапи. Останнє що нам залишилось, це наша статистика на сторінці “Dashboard”. Як написано у другому розділі, в нас є два графіки, перший – це статистика по замовленням, другий – по типах піддонів. Щоб промалювати ці графіки обрано бібліотеку “recharts”. Recharts - це бібліотека для створення інтерактивних графіків і діаграм за допомогою React. Вона пропонує широкий спектр типів графіків, таких як лінійні, кругові, стовпчасті, область, бари та інші. Ми можемо налаштовувати різноманітні аспекти графіків, включаючи кольори, шрифти, осі, легенду та анімацію. Ця бібліотека розроблена спеціально для використання з React-фреймворком, що дозволяє легко інтегрувати графіки в React-проекти, це як раз те що нам потрібно. У лістингу 3.11 модно побачити реалізацію одного із графіків, а саме статистика по типах піддонів. Також на рисунку А4 проілюстровано як виглядає статистика у нашому додатку

Лістинг 3.11 – Реалізація функціонала графіка піддонів

```
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip,
Legend, ResponsiveContainer } from 'recharts';
import { useTranslation } from 'react-i18next';
import Loader from 'components/Loader';
import NoData from 'components/NoData';
import { useAppSelector } from 'store/store';
const ChartNumberOfPallets = () => {
  const { t } = useTranslation();
  const { palletsLoading, palletsData } = useAppSelector((state) =>
state.dashboards);
  if (palletsLoading) return <Loader/>;
  if (!palletsData?.data.length) return <NoData/>;
  return ( <ResponsiveContainer width="100%" height={400}>
    <BarChart data={palletsData?.data}>
      <CartesianGrid strokeDasharray="3 3" />
      <XAxis dataKey="name" />
      <YAxis />
      <Tooltip />
      <Legend />
      <Bar dataKey="qty" barSize={75} name="Quantity"
fill="#8884d8" />
    </BarChart>
  </ResponsiveContainer>
```

```
);
};
export default ChartNumberOfPallets;
```

Отже, усі вимоги до функціоналу та дизайну реалізовано, ми розробили усі сторінки, створили три ролі, дали для кожної свої доступи, та розібрали як працює створення та життєвий цикл замовлення, і на станок проілюстровали як виглядає статистика нашого додатку. Використання React.js, Nest.js та інших бібліотек та технологій, дало нам змогу розробити якісний, швидкий, корисний оптимізований додаток, яким буде зручно користуватися будь-якому користувачеві. Усі технології довели свою ефективність та використовуються в широкому спектрі проектів. Результати роботи можуть бути використані для подальшого розвитку та покращення функціоналу додатку з метою задоволення потреб користувачів.

3.3 Аналіз результатів тестування

Перейдемо до тестування нашого коду. Для тестування коду нашої транспортної вебплатформи буде використовуватись Jest-testing. Jest є популярним фреймворком для тестування JavaScript-коду, зокрема для тестування React-проектів. Навіщо взагалі тестувати код? Тестування коду дозволяє виявити помилки та дефекти в програмному коді до того, як вони потраплять до кінцевого користувача. Це допомагає забезпечити якість продукту та підвищити задоволеність користувачів. Також тестові набори дають нам впевненість в тому, що зміни, які ми можемо внести в код (наприклад, рефакторинг), не ламають наявну функціональність. Для того щоб запустити тестування, є спеціальна команда, яка була заздалегідь додана до нашого package.json файлу, називається `npm run test`.

– найкращі практики - аналізує веб-сторінку на предмет дотримання найкращих практик розробки. Він перевіряє правильне використання HTML-тегів, уникнення блокуючого JavaScript на першому етапі завантаження, використання безпечних протоколів, оп

– SEO - перевіряє веб-сторінку на оптимізацію для пошукових систем. Він аналізує наявність метатегів, унікальність заголовків, використання правильних URL-адрес, структуровані дані та інші фактори, які впливають на SEO. За допомогою Lighthouse можна отримати рекомендації щодо покращення показників пошукової оптимізації.

Нижче на рисунку 3.12 показано результат тестів Lighthouse

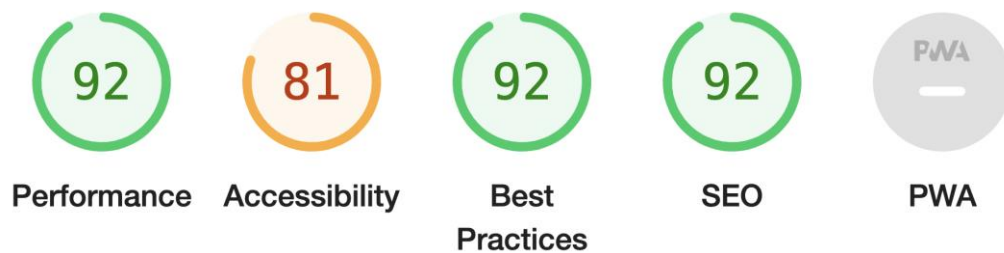


Рисунок 3.12 – Аналіз тестів Lighthouse

ВИСНОВКИ

В процесі виконання роботи, усі задачі були виконані. Розглянуто основні поняття та визначення «транспортна логістика», оптимізації транспортних систем, вибір виду і типу транспортних засобів, що виконуються за допомогою процесу планування, контролю за всіма транспортними та іншими операціями, наданням відповідної інформації вантажовласникам та здійснюється з мінімальними витратами. Також проаналізовано обрані бібліотеки, які використовувались для розробки додатку, розглянуто їх переваги над іншими аналогами.

У дипломній роботі розглянуто основні кроки розробки веб-додатку для логістики, використовуючи передові технології та бібліотеки. Були визначені вимогами до системи та розроблена архітектура системи.

При розробці, було обрано базу даних, розібрана взаємодія між користувачем та API. Розробили дизайн за допомогою онлайн-інструменту Figma.

У рамках даної дипломної роботи було проведено розробку функціоналу з використанням двох популярних технологій - React.js та Nest.js. Основною метою було створення ефективного та високопродуктивного веб-додатку з модульною архітектурою, що забезпечує зручну розробку, масштабованість та підтримку. Результатом роботи став розроблений функціонал, який успішно інтегровано в загальну структуру додатку. Використання React.js дало змогу забезпечити швидку реакцію на дії користувачів, підтримку різноманітних браузерів та платформ, а також забезпечити безпеку обробки даних. Компоненти та модулі були розроблені з врахуванням принципів модульності та повторного використання коду, що сприяє зручній розробці та розширенню функціоналу в майбутньому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Чепель Н.В. Моніторинг та логістика виробничих процесів галузі, курс лекцій для студ. спеціальності 7.05170108 «Технології зберігання, консервування та переробки молока» денної та заочної форм навчання. – К.: НУХТ, 2014. – 126с.
2. Маловичко А.С. Інтеграція України до транспортно-логістичної системи Європейського Союзу. Причорноморські економічні студії. 2016. Вип. 7. С. 55–58.
3. Тридід О.М. Логістика: навч. посіб. К.: Знання. - 2008. С. 566
4. Смирнов І.Г. Транспортна логістика: навч.пос. – К.: Центр учбової літератури. - 2008. – 224 с.
5. Сокур І.М. Транспортна логістика : навч. посіб. для студ. вищ. навч. закл. / І.М. Сокур, Л.М. Сокур, В.В. Герасимчук. – К.: Центр учбової літератури, 2009. – 222 с.
6. Пономарьова Ю.В. Логістика : навч. посіб. / Ю.В. Пономарьова. – 2-ге вид., переробл. та доп. – К.: Центр навчальної літератури, 2005. – 328 с.
7. Лифар В. В. Розвиток транспортної логістики в регіональній системі обслуговування товарних потоків. Маркетинг і менеджмент інновацій. 2017. № 4. С. 176-187.
8. Сухорукова Т.Г., Назаренко І.Л. Характеристика класифікаційних ознак транспортної логістики. Вісник економіки і промисловості. - 2014. – Вип. 47. – С. 51-57.
9. Корнієцький О. Є. Поняття та сутність логістики. Збірник наукових праць Агросвіт. 2015. №14. С. 1-4.
10. Резнік Н.П., Руденко С.В., Пилипчук К. М. Основні характеристики поняття логістики і системи управління ланцюгами постачань. InnovationandSustainability. 2022. No3. С. 95-102.

11. Конспект лекцій з дисципліни «Логістика» для здобувачів першого (бакалаврського) рівня вищої освіти зі спеціальності 073 «Менеджмент» / Укладач: С.І. Тарасенко. – Кам'янське: ДДТУ, 2020. – 144 с.
12. Фурман Б., Гапчак Т. Транспортна логістика та її основні проблеми. URL: <http://repository.vsau.org/getfile.php/4208.pdf> (дата звернення: 10.04.2023).
13. Anderson B. Transportation and trade across international borders / Bill Anderson, Juan Carlos Villa // Research in Transportation Business & Management. – Volume 16, September 2015, P. 1-3.
14. Брагинський В. В. Розвиток транспортно-логістичної системи як форма реалізації транзитного потенціалу України В. В. Брагинський. URL: <http://www.academy.gov.ua/ej/ej14/txts/Braginskiy.pdf> (дата звернення: 10.04.2023)
15. Дикань В. Л. Подходы к управлению инновационным развитием железнодорожного транспорта Украины / В. Л. Дикань, Ю. А. Рыжова // Вісник економіки транспорту і промисловості. – 2014. – Вип. 48. – С. 55-59. URL: http://nbuv.gov.ua/j-pdf/Vetp_2014_48_12.pdf (дата звернення 11.04.2023).
16. Дороховський О.М. Проблеми та перспективи розвитку транспортно-логістичної системи України / О.М. Дороховський //Економіка та управління. – 2012. - №5. – С. 60-65.
17. Крикавський Є. В. Сучасна логістика потребує інновацій Є. В. Крикавський. – URL: <http://essuir.sumdu.edu.ua/bitstream/123456789/28173/1/logist.pdf>.
18. Транспортно-логістична інфраструктура України: проблеми та перспективи розвитку/ І. Г. Лебідь, В. В. Кутирєв // Управління проектами, системний аналіз і логістика. – К.: НТУ, 2012. Вип. 10. С. 192–198.
19. Сирийчик Т. Транспортна політика України та її наближення до норм Європейського Союзу Т. Сирийчик та ін. ; за ред. Марчіна Свенціцькі. – К.: Аналіт.-дорадч. центр Блакитної стрічки, 2010. – 102 с. URL:http://www.undp.org.ua/files/en_76033Transport_System_Reform_Jun2010.pdf (дата звернення 11.04.2023).

20. Конспект лекцій з дисципліни «Логістика» для здобувачів першого (бакалаврського) рівня вищої освіти зі спеціальності 073 «Менеджмент» / Укладач: С.І. Тарасенко. – Кам'янське: ДДТУ, 2020. – 144 с.
21. Кислий В.М., Біловодська О.А., Олефіренко О.М., Соляник О.М. Л 69 Логістика: Теорія та практика: Навч. посіб. – К: Центр учбової літератури, 2010. – 360 с.
22. Соколова О. Є. Теоретико-методологічні основи формування транспортно-логістичної системи України. URL: <http://ecobio.nau.edu.ua/index.php/PPEI/article/viewfile/182/173> (дата звернення 11.04.2023).
23. Попова Н. В. Транспортно-логістична система: дефініція та складові // Бізнес Інформ. – 2016. - №1. С. 169-174. URL: http://nbuv.gov.ua/UJRN/binf_2016_1_27 (дата звернення 12.04.2023).
24. Конспект лекцій з дисципліни «Логістика» для здобувачів першого (бакалаврського) рівня вищої освіти зі спеціальності 073 «Менеджмент» / Укладач: С.І. Тарасенко. – Кам'янське: ДДТУ, 2020. – 144 с.

ДОДАТОК А

Таблиця А.1 – Ціни перевозки піддонів залежно від типу та області куди доставлятимуть

Назва областей	Ціна сімейства піддонів в грн							
	mini	quarter	half	medium	large	full	big	mega
Kyivska oblast	600	1000	1400	1800	2200	2600	3000	3400
Volynska oblast	960	1360	1760	2160	2560	2960	3360	3760
Dnipropetrovska oblast	800	1200	1600	2000	2400	2800	3200	3600
Donetska oblast	900	1300	1700	2100	2500	2900	3300	3700
Zhytomyrska oblast	700	1100	1500	1900	2300	2700	3100	3500
Zakarpatska oblast	1200	2560	3160	3760	4360	5960	6560	7160
Zaporizka oblast	1000	1400	1800	2200	2600	3000	3400	3800
Ivano-Frankivska oblast	1200	1600	2000	2400	2800	3200	3600	4000
Vinnytska oblast	900	1300	1700	2100	2500	2900	3300	3700
Kirovohradska oblast	900	1300	1700	2100	2500	2900	3300	3700
Luhanska oblast	900	1300	1700	2100	2500	2900	3300	3700
Lvivska oblast	1200	1600	2000	2400	2800	3200	3600	4000
Mykolaivska oblast	1000	1400	1800	2200	2600	3000	3400	3800
Odeska oblast	1160	1560	1960	2360	2760	3160	3560	3960
Poltavska oblast	800	1200	1600	2000	2400	2800	3200	3600
Rivnenska oblast	1160	1560	1960	2360	2760	3160	3560	3960
Sumska oblast	800	1200	1600	2000	2400	2800	3200	3600
Ternopil'ska oblast	1200	1600	2000	2400	2800	3200	3600	4000
Kharkivska oblast	900	1300	1700	2100	2500	2900	3300	3700
Khersonska oblast	1000	1400	1800	2200	2600	3000	3400	3800
Khmelnyska oblast	1160	1560	1960	2360	2760	3160	3560	3960
Cherkaska oblast	700	1100	1500	1900	2300	2700	3100	3500
Chernivetska oblast	1200	1600	2000	2400	2800	3200	3600	4000
Chernihivska oblast	700	2260	2860	3460	4060	4660	5260	5860

ДОДАТОК Б

Лістинг Б.1 – Форма створення замовлення

```
import { useEffect, useMemo } from 'react';
import { useNavigate } from 'react-router-dom';
import { Form, Row, Col, Input, DatePicker, Select } from 'antd';
import classNames from 'classnames';
import * as _ from 'lodash';
import BackButton from 'components/BackButton';
import CustomButton from 'components/CustomButton/CustomButton';
import { AdminPalletsTable } from 'components/EditableTable/Tables';
import { requestGetUserPriceList } from
'store/redux_slice/userPriceListSlice';
import GooglePlacesComponent from
'components/GooglePlacesComponent';
import { useParceAddress } from 'hooks/useParceAddress';
import { useAppDispatch, useAppSelector } from 'store/store';
import { setOrderData, clearOrderData, createOrderAction } from
'store/redux_slice/shippingSlice';
import { requestSearchUsers } from
'store/redux_slice/users/usersSearchSlice';
import { OrderType } from 'store/types';
import style from './newShipmentDeliveries.module.css';
const { Option } = Select;
const NewShipmentForm = () => {
  const [form] = Form.useForm();
  const navigate = useNavigate();
  const dispatch = useAppDispatch();
  const orderData = useAppSelector((state) => state.orderData);
  const usersSearch = useAppSelector((state) =>
state.usersSearch.users);
  const [parceAddressHandler] = useParceAddress();
  const palletsPriceSum = useMemo(() =>
orderData?.pallets?.reduce((prev, curr) => prev + curr.price, 0) ||
0, [orderData]);
  const palletsQtuSum = useMemo(() => (
    orderData?.pallets?.reduce((prev, curr) => prev +
curr.item_quantity, 0) || 0
  ), [orderData]);
  const palletsWeightSum = useMemo(() => (
    orderData?.pallets?.reduce((prev, curr) => prev + curr.weight,
0) || 0
  ), [orderData]);
  useEffect(() => {
    dispatch(requestSearchUsers(''));
  }, [dispatch]);
  useEffect(() => () => {
    dispatch(clearOrderData());
  }, [dispatch]);
```

```

const commitForm = _.debounce(() => {
  const values = form.getFieldsValue();
  const data = {
    pickup_address: {
      location: values?.pickup_address?.address?.location,
      company_name: values.from_name,
    },
    email: values.email,
    phone: values.phone,
    start_shipment_date: values.start_shipment_date ?
values.start_shipment_date.toJSON() : null,
  };
  dispatch(setOrderData(data));
}, [500]);
const proceed = () => dispatch(createOrderAction(orderData));
const onChangeClient = _.debounce((email: string) => {
  dispatch(requestGetUserPriceList(email));
}, 500);
const onChangeFieldAddress = _.debounce(async (fieldAddress) => {
  if (fieldAddress.placeId) {
    const { address } = await
parceAddressHandler(fieldAddress.fullAddress);
    form.setFieldsValue({
      pickup_address:{...fieldAddress,address:{location:address}},
    });
    dispatch(setOrderData({...orderData,
      pickup_address:{...orderData?.pickup_address,location:
address},
    }));
  }
}, 500);
const updateOptionHandler = (pallets) => {
  dispatch(setOrderData({...orderData,pallets}));
};
return (
  <>
    <div className={style.new_shipment_container_header}>
      <BackButton onClick={() => navigate(-1)} />
      <div className={classnames(style.title, 'main-title-
30px')}>New order</div>
    </div>
    <Form layout="vertical" className={style.new_shipment_form}
form={form} onFinish={proceed}>
      <div className={style.new_shipment_container}>
        <Form.Item label="Customer" name="email" rules={[{
required: true, message: 'User is required' }]}>
          <Select showSearch placeholder="Select user"
optionFilterProp="children" onChange={onChangeClient} >
            {usersSearch?.map((u) => <Option key={u.email}
value={u.email}>{u.name || u.email}</Option>)}
          </Select>
        </Form.Item>

```

```

        <div className={classnames(style.new_shipment_form,
'title-20px')}>Details</div>
        <Row className={style.details_part}>
            <Col span={11}>
                <Form.Item label="Indicate the place of collection of
goods" name="from_name" rules={[{ required: true, message: 'Name is
required' }]}>
                    <Input placeholder="Enter name"/>
                </Form.Item>
            </Col>
            <Col span={11} offset={2}>
                <Form.Item name="pickup_address" rules={[{ required:
true, message: 'Address is required' }]} >
                    <GooglePlacesComponent placeholder={'Enter city and
address'} address={'pickup_address'} setAddress={({pickupAddress}) =>
{ form.setFieldsValue({ pickup_address: pickupAddress })};
onChangeFieldAddress(pickupAddress);}}
                    />
                </Form.Item>
            </Col>
        </Row>
        <Row>
            <Col span={5}>
                <Form.Item label={'Pickup date'}
name="start_shipment_date" className={style.start_shipment_date_top}
rules={[{ required: true, message: 'Pickup date is required' }]}>
                    <DatePicker className="date_input"
disabledDate={({current}) => (current && current.valueOf() < new
Date().setDate(new Date().getDate() - 1))}format="DD/MM/YYYY"/>
                </Form.Item>
            </Col>
        </Row>
    </div>
    <div className={style.shipping_information_container}>
        <div className={style.cost_container}>
            <p className={style.cost_text}>
                Total cost: <span
className={style.price_text}>{`$${palletsPriceSum}uah`}</span>
            </p>
            <div className={style.options_container}>
                <p className={style.options_text}>
                    № pallets: <span
className={style.options_bolt_text}>{`$${palletsQtuSum}`}</span>
                </p>
                <p className={style.options_text}>
                    Weight: <span
className={style.options_bolt_text}>{`$${palletsWeightSum}kg`}</span>
                </p>
            </div>
        </div>
    </div>
    <AdminPalletsTable
        dataArray={orderData?.pallets || []}

```

```

    updateOptionHandler={updateOptionHandler}
    address={orderData?.pickup_address}
  />
  <div className={style.new_shipment_container}>
    <Form.Item
      label={<div className={style.note_style}>Note</div>}
      name="note"
    >
      <Input.TextArea placeholder="Entry any notes" />
    </Form.Item>
    <div className={style.buttons_container}>
      <CustomButton
        type="primary"
        className={style.proceedBtn}
        htmlType="submit"
      >
        Confirm
      </CustomButton>
      <CustomButton
        className={classnames('defaultBtn', style.cancelBtn)}
        onClick={() => {
          navigate(-1);
          form.resetFields();
        }}
      >
        Cansel
      </CustomButton>
    </div>
  </div>
</Form>
</>
);
};
export default NewShipmentForm;

```

ДОДАТОК В

Лістинг В.1 – Реалізація функціоналу мапи, за допомогою Mapbox

```
import { useState, useEffect, useMemo } from 'react';
import { Checkbox } from 'antd';
import cn from 'classnames';
import axios from 'axios';
import ReactMapGL, { Source, Layer, Marker } from 'react-map-gl';
import SuccessIconSvg from 'assests/svg/SuccessIconSvg';
import { TrucksStructure } from 'store/types';
import style from './map.module.css';
import 'mapbox-gl/dist/mapbox-gl.css';
type RoadsTypes = { color: string, routes: number[][] | [],
locationStr: string, visible: boolean, truckNumber: string,
driverName: string, markers: {date_end: boolean, lng: number, lat:
number}[],
}[]
const stockLocationKyiv = { lat: 50.397695, lng: 30.528378 };
const MapboxContainer = ({ trucks }: { trucks: TrucksStructure[] })
=> {
  const [visible, setVisible] = useState(false);
  const [roads, setRoads] = useState<RoadsTypes>([]);
  const mapboxMarkers = useMemo(() => (
    trucks?.map((truck) => {
      const locationStr = truck.trips.map((trip) => (
        `${trip.shipping_request.pickup_address.location.lng}%2C${
          trip.shipping_request.pickup_address.location.lat}%3B${
            stockLocationKyiv.lng}%2C${stockLocationKyiv.lat}`
      )).join('%3B');
      const markers = truck.trips.map((trip) => ({
        date_end: !!trip.date_end,
        lng: trip.shipping_request.pickup_address.location.lng,
        lat: trip.shipping_request.pickup_address.location.lat,
      }));
      markers.push({ date_end: false, ...stockLocationKyiv });
      return { color: truck.color, locationStr, visible: true,
truckNumber: truck.plate_number, driverName: truck?.driver?.name ||
'-' , markers };
    })
  ), [trucks]);
  const toggleDrawer = () => { if (visible) {setVisible(false) }
else {setVisible(true)}
};
  useEffect(() => {
    if (mapboxMarkers && mapboxMarkers?.length > roads.length) {
      mapboxMarkers?.forEach(async (item, i) => {
        let roadsArr: number[][] = [];
        if (item.markers.length > 1) {
          const { data } = await
axios.get(`https://api.mapbox.com/directions/v5/mapbox/driving/${ite
```

```

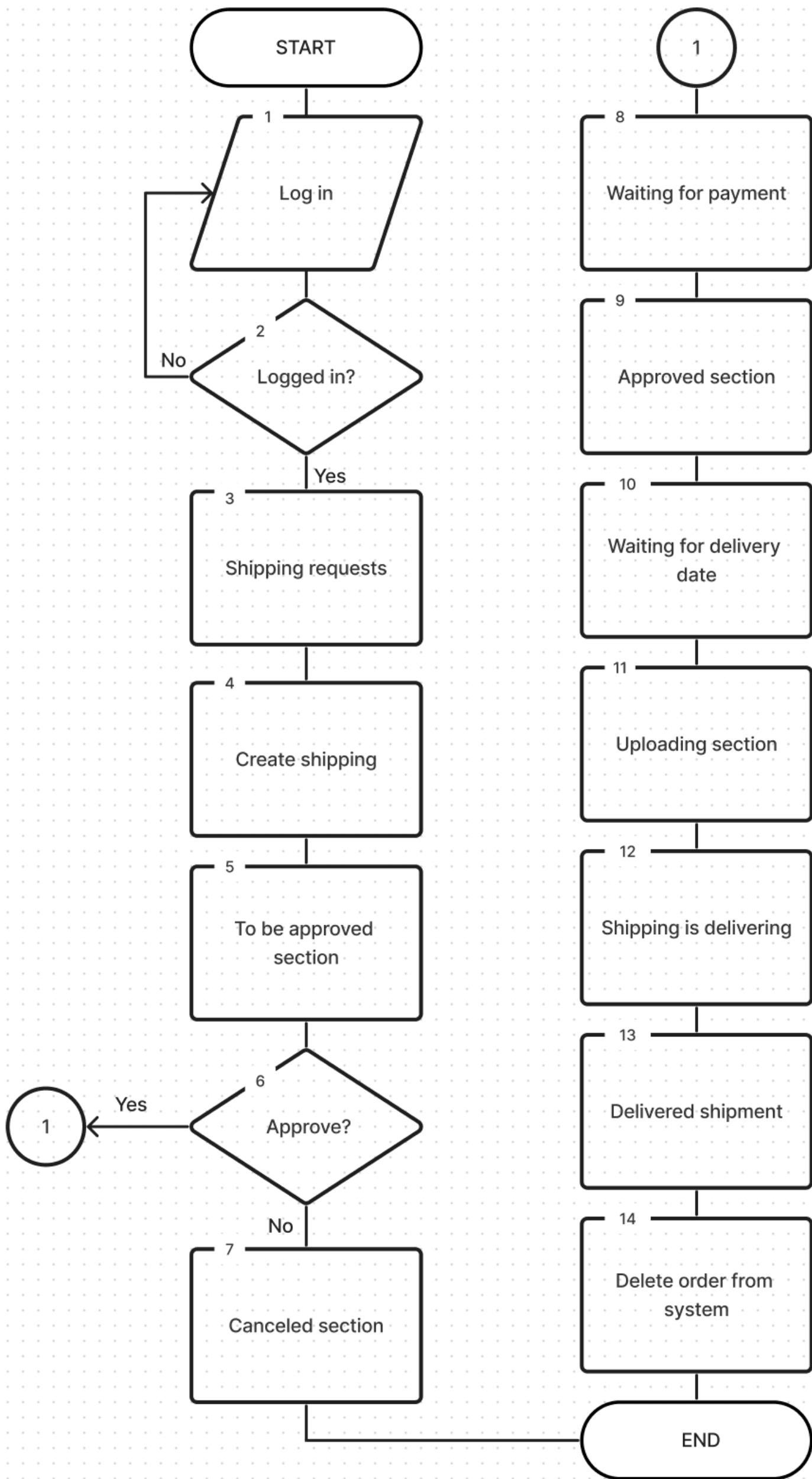
m.locationStr}?alternatives=true&geometries=geojson&language=en&over
view=full&steps=true&access_token=${process.env.MAPBOX_TOKEN}`);
    if (data.code === 'Ok' && data.routes.length) {
        const [route] = data.routes;
        roadsArr = route.geometry.coordinates;
    }
    }
    setRoads((prevState) => [...prevState, { routes: roadsArr,
...item }]);
    });
    }
    }, [mapboxMarkers]);
    const checkboxOnChange = (e) => {
        setRoads((prevState) => {
            const newState = prevState.map((obj) => {
                if (obj.truckNumber === e.target.value) {
                    return { ...obj, visible: !obj.visible };
                }
                return obj;
            });
            return newState;
        });
    };
    if (!mapboxMarkers) return null;
    return (
        <>
        <div className={cn({ [style.drawer_is_open_mask]: visible
    })}></div>
        <div className={cn(style.drawer_map_container, {
            [style.drawer_is_open]: visible,
        })}>
            <div className={style.btn_container}>
                <button className={style.drawer_btn}
onClick={toogleDrawer}>
                    { visible ? 'Nascondi mappa percorsi' : 'Vedi mappa
percorsi' }
                </button>
            </div>
            {roads.length ? (
                <div className={style.routes_board}>
                    <div className={style.routes_board_container}>
                        { roads.map((m, i) => (
                            <Checkbox key={i} className={style.checkbox_item}
value={m.truckNumber} checked={m.visible}
onChange={checkboxOnChange}
                            >
                                <div className={style.item_content}>
                                    <span className={style.round_color} style={{
backgroundColor: m.color }}></span>
                                    <div className={style.board_info}>
                                        <span
className={style.truck_number}>{m.truckNumber}</span>
                                        <span>{m.driverName}</span>

```

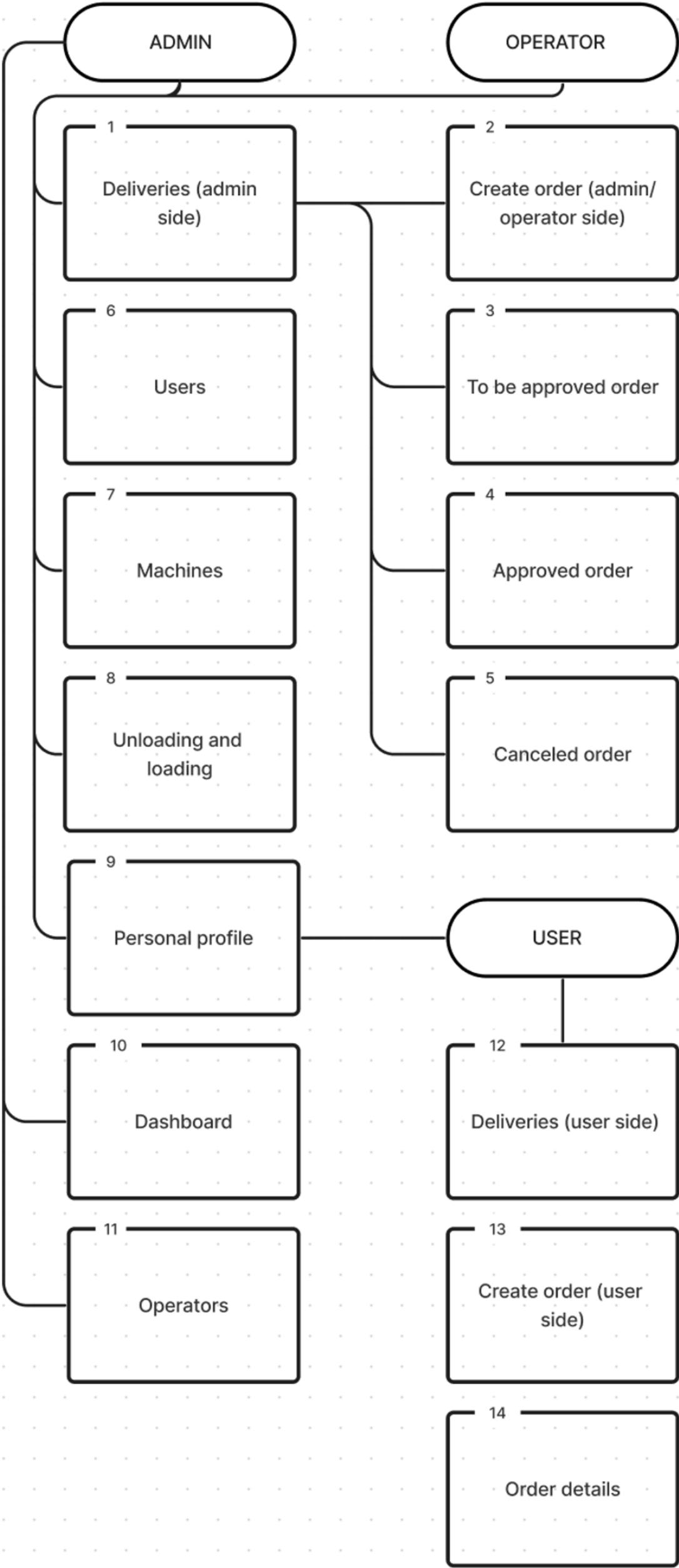
```

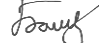
        </div>
      </div>
    </Checkbox>
  )))
</div>
</div>
) : null}
<div className={cn(style.map_container, { [style.blur]:
!visible })}>
  <ReactMapGL initialViewState={{latitude: 48.3358856,
longitude: 31.1788196, zoom: 5 }}
mapStyle="mapbox://styles/trasportipesanti/cl7zwu5pw00av14mpss83p70n
" mapboxAccessToken={process.env.MAPBOX_TOKET}>
    { roads.map((item, i) => item.visible && (
      <Source key={i} type="geojson" data={{
        type: 'FeatureCollection',
        features: [{
          type: 'Feature',
          properties: {},
          geometry: {
            type: 'LineString',
            coordinates: item.routes,
          },
        }],
      }}>
        <Layer type="line" layout={{ 'line-join': 'round' }}
paint={{ 'line-color': item.color, 'line-width': 5 }} />
      </Source>
    ))}
    { roads.map((road) => road.visible && (
      road.markers.map((trip, i) => (
        <Marker key={i} longitude={trip.lng}
latitude={trip.lat}>
          <div style={{ backgroundColor: road.color }}
className={style.markers}>
            { trip.date_end && <SuccessIconSvg color="#fff"
/> }
          </div>
        </Marker>
      ))
    ))}
  </ReactMapGL>
</div>
</div>
</>
);
};
export default MapboxContain

```



					13.02070849.00001 A1					
					Розробка транспортної логістичної вебплатформи з використанням React.js	Лист.			Маса	Масшт.
Зам.	Лист.	№ докум.	Підпис	Дата						
Розробив		Балабуха О.М.								
Перевірів		Киричек Г.Г.								
					Алгоритм роботи процесу замовлення	Лист.			Лист.	
Н. контр.		Щербак Н.В.				НУ “Запорізька політехника” КНТЗ-519				
Затвердж.		Кудерметов Р.К.								



					13.02070849.00001 ПЛ1					
					Розробка транспортної логістичної вебплатформи з використанням React.js Схема сторінок системи та доступ по ролям до них	Лист.		Маса	Масшт.	
Зам.	Лист.	№ докум.	Підпис	Дата						
Розробив		Балабуха О.М.								
Перевірів		Киричек Г.Г.								
Н. контр.		Щербак Н.В.				Лист.		Лист.		
Затвердж.		Кудерметов Р.К.				НУ “Запорізька політехника” КНТЗ-519				

User side

Deliveries (user side)

TRUCK LOGO

Deliveries

Receiver name

Pickup date

Reset

New order

☐	Id	N°pallet	Receiver name	Pickup date	Status	
☐	1344	3	Name	12-10-2021	Awaiting approval	...
☐	1344	3	Name	12-10-2021	Awaiting payment	...
☐	1344	3	Name	12-10-2021	Awaiting collection	...
☐	1344	3	Name	12-10-2021	Delivering	...
☐	1344	3	Name	12-10-2021	Rejected	See details Delete

New order (user side)

TRUCK LOGO

Deliveries

BACK

NEW ORDER

Details

Indicate the place of collection of goods

Name

Address

Pickup date

Total cost 1500.00uah

N° pallet: 2 Weight: 338kg

Qty	Short side	Long side	Height	Weight	Prezzo	
2	120cm	220cm	110cm	120kg	40	...

Add pallets

Note

Enter any notes

Confirm

Cancel

Order details (user side)

TRUCK LOGO

Deliveries

BACK

ORDER #234

Delivering

Total cost: 60.00€

Address info: Name ADDRESS INFORMATION

N° pallets: 3 Weight: 330kg

Pickup date: 22-12-2021

Qty	Short side	Long side	Height	Weight
2	120cm	220cm	110cm	120kg
1	80cm	140cm	110cm	80kg

Personal profile

TRUCK LOGO

Deliveries

Profile

Logout

YOUR INFO

Name

Address

Email: xxxxx@xxxxx.com

Phone number: 3456789797

UPDATE PASSWORDS

Old Password

New Password

Retype new password

Aggiorna password

Admin side

Unloading and loading

TRUCK LOGO

Dashboards

Deliveries

Users

Machines

Unloading and loading

Operators

Unloading and loading 01-Dic-2021

Shipments to be associated

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

Vedi mappa percorsi

Unloading and loading (Map)

TRUCK LOGO

Dashboards

Deliveries

Nascondi mappa percorsi

Unloading and loading

Operators

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

XF5867FB Luca Spautani

To be approve/Canceled/Approved tables

TRUCK LOGO

Dashboards

Deliveries

Users

Machines

Unloading and loading

Operators

Approved

Approved orders

Receiver name

Pickup date

Reset

New order

Id	N°pallets	Client name	Approved date	Pickup date	Delivery date	Status	Driver
1344	3	Name	13-10-2021	13-10-2021	13-10-2021	Awaiting approval	Driver Name
1344	3	Name	13-10-2021	13-10-2021	13-10-2021	Awaiting approval	Driver Name
1344	3	Name	13-10-2021	13-10-2021	13-10-2021	Awaiting approval	Driver Name

Show details

Delete

To be approve/Canceled/Approved tables

TRUCK LOGO

Dashboards

Deliveries

Users

Machines

Unloading and loading

Operators

Approved

SPEDIZIONE #134

Approvato il: 27-12-2022

Stato: Delivering

Address info: Name address info

Machine: AK3244K

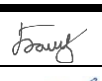
Driver: Driver name

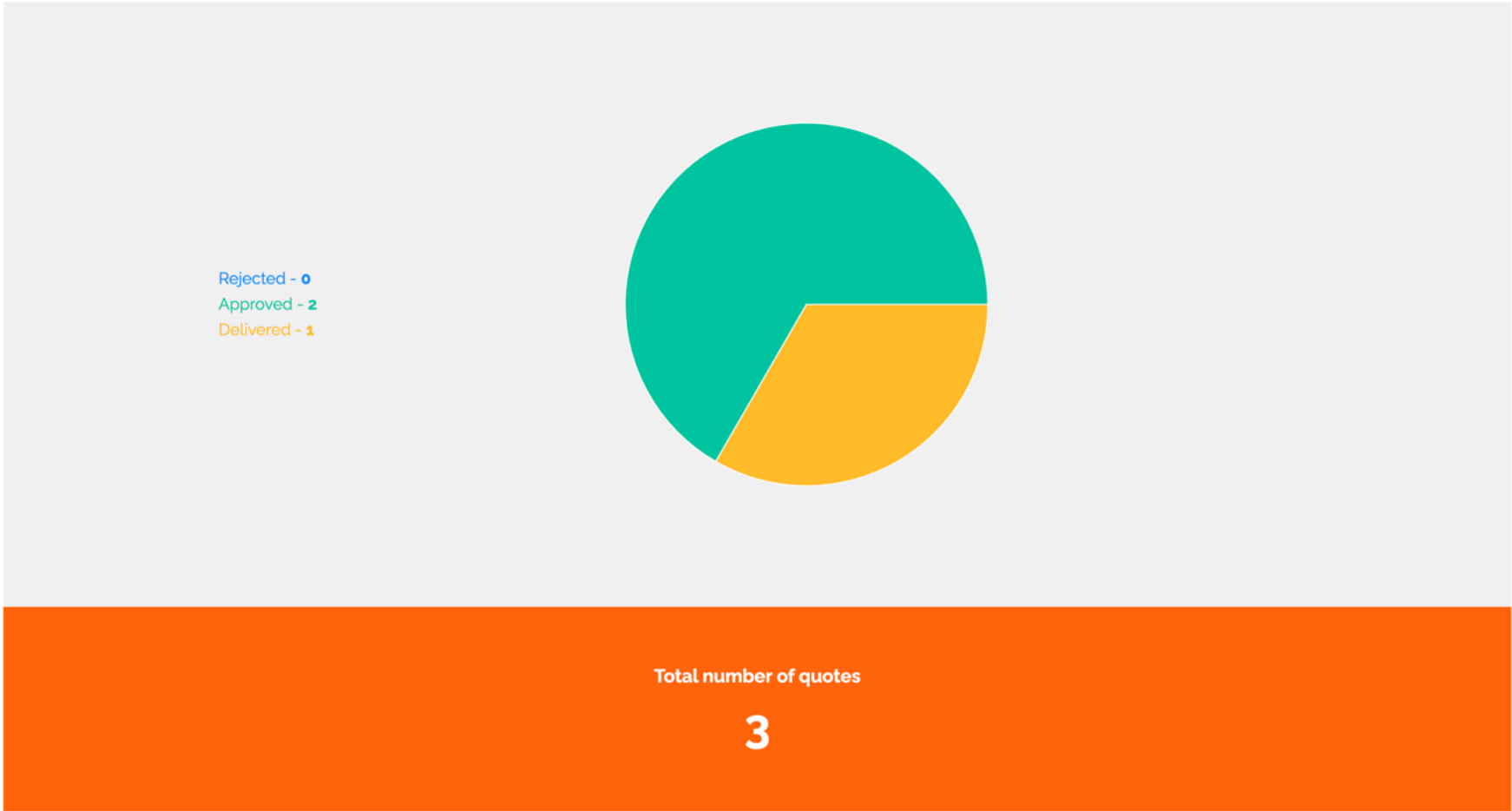
N° pallet: 3

Peso: 338kg

Qty	Short side	Long side	Height	Weight	Prezzo	
2	120cm	220cm	110cm	120kg	€ 20.00	*
1	80cm	140cm	110cm	80kg	€ 20.00	*

Note

					13.02070849.00001 ПЛ2				
					Розробка транспортної логістичної вебплатформи з використанням React.js	Лист.	Маса	Масшт.	
Зам.	Лист.	№ докум.	Підпис	Дата	Дизайн сторінок вебплатформи				
Розробив		Балабуха О.М.							
Перевірів		Киричек Г.Г.							
Н. контр.		Щербак Н.В.							
Затвердж.		Кудерметов Р.К.							
						Лист.	Лист.		
									НУ “Запорізька політехніка” КНТЗ-519



					13.02070849.00001 ПЛЗ						
					Розробка транспортної логістичної вебплатформи з використанням React.js	Лист.			Маса	Масшт.	
Зам.	Лист.	№ докум.	Підпис	Дата							
Розробив		Балабуха О.М.									
Перевірів		Киричек Г.Г.									
					Статистика системи	Лист.			Лист.		
Н. контр.		Щербак Н.В.				НУ “Запоризька політехника” КНТЗ-519					
Затвердж.		Кудерметов Р.К.									