

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

на тему:

**«Веб-платформа автозаправних станцій для продажу
пального»**

Виконав студент групи ІСТ-20-1
ДЗЮБЕНКО Дмитро Олегович

Керівник роботи:
ОРИНЧАК Андрій Іванович

Рецензент:
МАРЧУК Дмитро Костянтинович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
Марина ГРАФ
«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

Здобувач вищої освіти: ДЗЮБЕНКО Дмитро Олегович

Керівник роботи: ОРИНЧАК Андрій Іванович

Тема роботи: «Веб-платформа автозаправних станцій для продажу пального»,
затверджена наказом закладу вищої освіти від «25» січня 2024 р., №24/с

Термін здачі закінченої роботи: «10» червня 2024 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

об'єктом дослідження є веб-платформа автозаправних станцій для продажу пального.

Предметом дослідження виступає використання Web-орієнтованих технологій для
розробки веб-платформи автозаправних станцій для продажу пального.

Проект: <https://git.ztu.edu.ua/diploma2024/ist-20-1/dziubenko-dmytro-olehovych>

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1,2	Оринчак А.І.		

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних рішень та їх особливостей. Формулювання технічного завдання до проекту. Опрацювання літературних джерел.	2 лютого 2024 – 5 березня 2024	Виконано
2	Проектування структури інформаційної системи	6 березня 2024 – 25 березня 2024	Виконано
3	Написання програмного коду	26 березня 2024 – 26 квітня 2024	Виконано
4	Тестування системи	27 квітня 2024 – 14 травня 2024	Виконано
5	Оформлення пояснювальної записки	15 травня 2024 – 1 червня 2024	Виконано
6	Попередній захист	1 червня 2024	Виконано

РЕФЕРАТ

Кваліфікаційна робота складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 62 сторінці друкованого тексту.

Пояснювальна записка має 48 сторінок додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 26 рисунків. Загальний обсяг роботи – 110 сторінок.

Метою кваліфікаційної роботи є розробка та аналіз системи для продажу пального онлайн шляхом створення веб-додатку з використанням Web-орієнтованих технологій.

У роботі запропоновано використання інформаційних технологій для розробки та впровадження веб-платформи автозаправних станцій для продажу пального з метою оптимізації роботи управління.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-ДОДАТОК, CSS, JAVASCRIPT, HTML, BOOTSTRAP, NODE.JS, REACT, АЗС WEB-САЙТ, MVC, СЕРВЕР, CRUD, API, MERN.

					ІСТ.КР.Б – 126 – 02 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Веб-платформа автозаправних станцій для продажу пального Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	Д.ОДЗЮБЕНКО						3	110
Керівник	А.І.ОРИНЧАК					Житомирська політехніка, група ІСТ-20-1		
Рецензент	Д.К. МАРЧУК							
Н. контр.								
Зав. каф.	М.С. ГРАФ							

ABSTRACT

The qualification work consists of a list of abbreviations, an introduction, three chapters, conclusions, a list of references and appendices.

The text part is set out on 62 pages of printed text.

The explanatory note has 48 pages of appendices. The list of references includes 10 titles and occupies 1 page. The paper contains 26 figures. The total volume of the work is 110 pages.

The purpose of the qualification work is to develop and analyze a system for online fuel sales by creating a web application using web-oriented technologies.

The work proposes the use of information technologies to develop and implement a web platform for gas stations to optimize management operations.

KEYWORDS: INFORMATION SYSTEM, WEB APPLICATION, CSS, JAVASCRIPT, HTML, BOOTSTRAP, NODE.JS, REACT, GAS STATION WEBSITE, MVC, SERVER, CRUD, API, MERN.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП	8
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1 Аналіз проекту та постановка задачі	9
1.2 Аналіз аналогів програмного продукту	13
1.3 Вибір архітектури для побудови інформаційної системи	15
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
Висновки до першого розділу	23
РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ....	24
2.1 Варіанти використання системи	24
2.2 Розробка бази даних системи	25
2.3 Проектування архітектури інформаційної системи	31
2.4 Реалізація основного функціоналу системи.....	38
Висновки до другого розділу.....	46
РОЗДІЛ 3 ОГЛЯД ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	47
3.1 Огляд інтерфейсу системи	47
3.2 Тестування системи	58
Висновки до третього розділу	59
ВИСНОВОК.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61
ДОДАТКИ.....	62
Додаток А	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СКОРОЧЕНЬ І ТЕРМІНІВ

ІС — інформаційна система

СУБД — система управління базами даних

БД — база даних

MVC — model-view-controller, архітектурний шаблон, що використовується під час проєктування та розробки програмного забезпечення

ІТ — інформаційні технології

HTML — мова розмітки гіпертексту (HyperText Markup Language)

CRUD — створення, читання, оновлення, видалення (create, read, update, delete)

MERN — (MongoDB, Express.js, React.js, Node.js) є сучасною архітектурою для створення веб-додатків

API – (application programming interface) це спосіб завдяки якому дві або більше комп'ютерні програми можуть спілкуватися між собою

SPA – (single-page application) це вебзастосунок чи вебсайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою

PWA – (progressive web app) це різновид веб-додатків, які створюються на базі наявних сайтів і поєднують у собі властивості нативних мобільних додатків та браузерних можливостей сайтів

АЗС – автозаправна станція

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У нинішньому світі, в якому переважна більшість людей використовують автотранспорт для подорожей з пункту А в пункт Б, все більшу і більшу актуальність набувають системи, які дозволяють купляти паливо, не виходячи з авто. Тож впровадження систем дистанційного продажу палива є ключовим кроком у розвитку інфраструктури обслуговування клієнтів і підвищенні конкурентоспроможності на сучасному ринку.

Метою кваліфікаційної роботи є розробка та аналіз системи для продажу пального онлайн. Основне завдання полягає в створенні програмного додатку, який забезпечить змогу користувачів замовляти пальне онлайн, а постачальникам дасть змогу його продавати.

Постановлена мета дослідження зумовлює необхідність вирішення декількох завдань, а саме: вивчення потреб та вимог клієнтів; аналіз вже існуючих рішень, виокремлення їх переваг та недоліків; проектування оптимальної архітектури, яка забезпечить надійність та доступність роботи в ній для користувача.

Об'єктом дослідження є процес продажу та купівлі пального онлайн.

Предметом дослідження є використання інформаційних технологій для створення веб-додатку для продажу та купівлі пального онлайн.

Методи дослідження. Було проведено аналіз вимог користувачів і надавачів послуг, проведено аналіз вже існуючих рішень, спроектовано архітектуру системи та розроблено базу даних. Також проводилося тестування функціоналу додатку для забезпечення надійності системи та зручності її використання. Практичне значення полягає в тому, що результати роботи можуть бути використані: для оптимізації операційних витрат для компаній, підвищення безпеки операцій, підвищення ефективності бізнесу, підвищення зручності та доступності для клієнтів

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз проекту та постановка задачі

Створення веб-додатка для онлайн-продажу пального вимагає реалізації кількох базових функцій, які забезпечують зручність використання, ефективність і безпеку. Вибір мови JavaScript для цієї задачі має свої переваги, які я також розгляну.

Базові функції веб-додатка для онлайн-продажу пального

1. Реєстрація та аутентифікація користувачів

Функції:

Реєстрація нових користувачів: Створення облікових записів з базовими контактними даними.

Вхід: Перевірка облікових даних користувачів.

Скидання пароля: Можливість відновлення доступу у разі забутого пароля.

Причини:

Безпека: Захист даних користувачів і контроль доступу.

Індивідуальний підхід: Персоналізація досвіду для користувача.

2. Перегляд асортименту пального

Функції:

Каталог продукції: Відображення доступних видів пального з детальним описом, цінами та наявністю.

Фільтри і сортування: Можливість відбору продукції за різними критеріями (тип пального, ціна, тощо).

Деталі продукту: Інформація про склад, властивості і застосування пального.

Причини:

Зручність: Полегшення вибору продукту для користувача.

Інформативність: Забезпечення повної інформації про продукти для прийняття рішення.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		9

3. Кошик та управління замовленнями

Функції:

Додавання до кошика: Можливість додати пальне в кошик для подальшої покупки.

Редагування кошика: Зміна кількості, видалення або додавання товарів.

Перегляд замовлень: Список поточних і минулих замовлень користувача.

Причини:

Комфорт: Зручність управління покупками.

Організація: Просте оформлення замовлень і їх перегляд.

4. Оформлення та оплата замовлення

Функції:

Форма оформлення: Введення даних для доставки та оплати.

Інтеграція з платіжними системами: Підтримка різних способів оплати (банківські карти, електронні гаманці).

Підтвердження замовлення: Підтвердження та відправка квитанцій на електронну пошту.

Причини:

Зручність: Спрощення процесу покупки.

Безпека: Захист платіжних даних користувача.

5. Моніторинг та повідомлення

Функції:

Стан замовлення: Відстеження статусу замовлення (опрацьовується, доставлено, тощо).

Повідомлення: Сповіщення про зміни в статусі замовлення, акції, новини.

Причини:

Інформативність: Оперативне інформування користувача про важливі події.

Залучення: Підвищення залученості користувачів.

6. Профіль користувача

Функції:

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Редагування профілю: Можливість змінювати контактні дані, налаштування облікового запису.

Історія покупок: Перегляд попередніх замовлень і їх деталей.

Причини:

Зручність: Управління персональною інформацією.

Аналітика: Перегляд історії покупок для кращого розуміння потреб користувачів.

7. Пошук і зворотний зв'язок

Функції:

Пошук: Можливість швидкого пошуку пального по ключовим словам або фразам.

Контактна форма: Зворотний зв'язок для користувачів для запитань або скарг.

Причини:

Ефективність: Швидкий доступ до необхідної інформації.

Підтримка: Забезпечення комунікації з користувачами.

8. Аналітика і звітність

Функції:

Аналітика продажів: Звіти про продажі за періодами, популярні продукти.

Моніторинг активності користувачів: Аналіз поведінки користувачів на сайті.

Причини:

Оптимізація: Поліпшення асортименту і маркетингових стратегій.

Прогнозування: Підготовка прогнозів на основі даних.

Щодо обраної мови програмування для розробку веб-додатку, було обрано мову програмування JavaScript. Наведемо переваги використання JavaScript для розробки:

1. Єдина мова для фронтенда і бекенда

JavaScript використовується як для клієнтської, так і для серверної частини завдяки Node.js.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Переваги:

Скорочення кривої навчання: Одна мова для всіх частин додатка.

Повторне використання коду: Один і той самий код можна використовувати на сервері і клієнті.

Приклад: Моделі даних можуть бути написані один раз і використовуватися як в бекенді, так і у фронтенді.

2. Широка підтримка і екосистема

JavaScript має одну з найбільших екосистем з бібліотеками та фреймворками для різних задач.

Переваги:

Великий вибір інструментів: Швидка розробка завдяки використанню готових рішень.

Активна спільнота: Постійна підтримка та оновлення.

Приклад: Використання фреймворків, таких як Express.js для бекенду і React.js для фронтенду, значно спрощує розробку.

3. Висока продуктивність і асинхронність

JavaScript підтримує асинхронну модель виконання, що дозволяє ефективно обробляти одночасні запити.

Переваги:

Швидкодія: Асинхронні операції підвищують продуктивність додатка.

Масштабованість: Легше обробляти велику кількість запитів.

Приклад: Сервер може обробляти запити на замовлення пального і одночасно виконувати інші задачі без затримок.

4. Гнучкість і динамічний розвиток

JavaScript постійно розвивається і адаптується до нових вимог веб-розробки.

Переваги:

Актуальність: Можливість використовувати нові фічі і стандарти.

Гнучкість: Легкість у впровадженні нових функцій і технологій.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Приклад: Використання сучасних бібліотек і фреймворків, що з'являються у JavaScript екосистемі, забезпечує актуальність додатка.

5. Підтримка сучасних фронтенд технологій

JavaScript є основою для створення сучасних інтерфейсів користувача.

Переваги:

Підтримка SPA і PWA: Можливість створення сучасних односторінкових і прогресивних веб-додатків.

Реальні-time функціонал: Підтримка веб-сокетів і інших технологій для створення реальні-time додатків.

Приклад: Створення інтерактивного інтерфейсу для замовлення пального з динамічним оновленням цін і наявності.

6. Легка інтеграція з іншими сервісами і API

JavaScript має численні інструменти і бібліотеки для інтеграції з різними веб-сервісами і API.

1.2 Аналіз аналогів програмного продукту

Було проаналізовано декілька прикладів додатків для дистанційного продажу палива, виокремлено їх переваги та недоліки:

1. WOG PAY - Мобільний додаток від української мережі АЗС WOG, який дозволяє заправити автомобіль безпосередньо з мобільного телефону.

Переваги:

Швидкість: Можливість оплатити паливо безпосередньо через додаток, без необхідності відвідувати касу.

Зручність: Можливість обирати вид палива і кількість літрів заздалегідь.

Інтеграція з програмою лояльності: Клієнти можуть накопичувати бонуси та отримувати знижки.

Недоліки:

Залежність від інтернету: Необхідність наявності стабільного інтернет-з'єднання.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Обмеженість функціоналу: Додаток може не підтримувати всі види платежів або акцій.

2. Wayne Fueling Systems - Рішення від компанії Wayne, яке надає можливість безконтактної оплати палива через мобільний додаток.

Переваги:

Інтеграція з POS-системами: Підтримує синхронізацію з існуючими системами продажу на АЗС.

Безпека: Використовує сучасні технології для захисту даних клієнтів.

Недоліки:

Вартість впровадження: Для АЗС може бути необхідним оновлення обладнання.

Залежність від конкретного постачальника: Може обмежувати вибір обладнання та послуг.

3. Shell App - Мобільний додаток від Shell, який дозволяє клієнтам заправлятися, здійснювати платежі, а також отримувати бонуси та знижки.

Переваги:

Широкий функціонал: Підтримка різних видів платежів, доступ до акцій і програм лояльності.

Легкість використання: Інтуїтивний інтерфейс для зручності користувачів.

Недоліки:

Регіональні обмеження: Не всі функції доступні в кожній країні.

Необхідність реєстрації: Потрібно створити обліковий запис для використання всіх функцій.

4. Mobile Fueling Apps (загальні, наприклад, Booster Fuels або Yoshi) - Це додатки, які надають послуги заправки «на місці», коли паливо доставляється безпосередньо до автомобіля клієнта, будь-то вдома чи в офісі.

Переваги:

Комфорт: Водії можуть заправляти автомобіль без необхідності відвідувати АЗС.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Часова економія: Ідеально підходить для людей з щільним графіком.

Недоліки:

Вартість: Послуги можуть бути дорожчими за традиційне заправлення.

Доступність: Обмежено певними географічними районами.

5. Puma Energy Fuel Station Locator - Додаток від Puma Energy, який допомагає знайти найближчі АЗС, а також дозволяє оплачувати паливо і отримувати бонуси.

Переваги:

Легкість навігації: Карта для пошуку найближчих АЗС.

Інтеграція з бонусними програмами: Підтримка акцій і програм лояльності.

Недоліки:

Обмежена функціональність: Основний акцент на навігацію, а не на дистанційну оплату.

Необхідність оновлень: Потребує регулярного оновлення для відображення актуальної інформації про АЗС.

Розвиток додатків для дистанційного продажу палива покращує взаємодію між клієнтами та постачальниками, робить процес заправки більш сучасним і зручним, але потребує адаптації з боку користувачів та бізнесів.

1.3 Вибір архітектури для побудови інформаційної системи

Стосовно архітектури, яку було обрано для побудови веб-додатку, було обрано MVC. Model-View-Controller (MVC) є популярною архітектурною схемою для розробки програмного забезпечення, особливо у веб-додатках. Її актуальність пояснюється рядом переваг, які надають ефективність і гнучкість в процесі розробки, підтримки, і масштабування програм.

Основи MVC:

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		15

- Model (Модель): Представляє дані програми та логіку бізнес-процесів. Вона відповідає за управління даними і реагує на запити з боку Контролера, надаючи дані у вигляді, необхідному для Виду.

- View (Вид): Відповідає за відображення даних, що надходять від Моделі, і надає інтерфейс користувача. Вид також оновлюється у відповідь на зміни в Моделі.

- Controller (Контролер): Опрацьовує запити користувачів і взаємодіє з Моделлю для оновлення даних. Він також визначає, який Вид повинен бути представлений користувачу на основі результатів взаємодії з Моделлю.

В чому полягає ктуальність MVC:

1. Розділення обов'язків

MVC розділяє логіку бізнесу, інтерфейс користувача та управління користувацькими запитами на окремі компоненти.

Переваги:

- Чистіший код: Легше зрозуміти і підтримувати, оскільки компоненти виконують чітко визначені функції.

- Модульність: Розробники можуть працювати над Моделлю, Видом або Контролером незалежно один від одного.

Приклад: Команда розробників може паралельно працювати над новими інтерфейсами користувача (View) та вдосконаленням алгоритмів бізнес-логіки (Model), що підвищує ефективність.

2. Полегшення тестування

Кожен компонент MVC може бути протестований окремо, що спрощує виявлення та виправлення помилок.

Переваги:

- Юніт-тестування: Моделі та Контролери можуть бути протестовані незалежно, що сприяє кращій надійності програмного забезпечення.

- Інтеграційне тестування: Тестування інтеграції компонентів проводиться з більшою впевненістю, оскільки кожен з них був протестований окремо.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Приклад: Тестування складних бізнес-логік у моделі не потребує взаємодії з інтерфейсом користувача, що спрощує процес.

3. Повторне використання компонентів

MVC дозволяє повторно використовувати Моделі та Види в різних частинах програми або навіть в інших додатках.

Переваги:

- Ефективність: Знижує зусилля на розробку нових компонентів, використовуючи вже існуючі.
- Гнучкість: Легко створювати нові інтерфейси або додавати нову функціональність, повторно використовуючи вже створені компоненти.

Приклад: Створений раніше View може бути використаний для відображення тих самих даних у різних частинах програми.

4. Легкість модифікацій і масштабування

Розділення на компоненти дозволяє швидко вносити зміни в будь-який з компонентів, не зачіпаючи інші.

Переваги:

- Зручність змін: Легко адаптуватися до нових вимог або вдосконалювати окремі частини програми.
- Масштабованість: Додаючи нові функції, можна легко інтегрувати їх в існуючу структуру.

Приклад: Додавання нової функції, як-от додаткове поле у формі, може вимагати лише зміни View і мінімальних змін в Controller.

5. Підтримка асинхронних процесів

MVC дозволяє легко інтегрувати асинхронні запити та обробку даних.

Переваги:

- Реактивність: Програми можуть реагувати на дії користувачів і змінювати інтерфейс без необхідності перезавантаження сторінок.
- Покращена продуктивність: Знижується навантаження на сервер, оскільки обробка може бути розподілена.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Приклад: Асинхронне завантаження даних у таблицю дозволяє користувачу продовжувати взаємодію з програмою без очікування завершення всіх запитів.

6. Підтримка різних технологій і мов програмування

MVC використовується в різних мовах і фреймворках, таких як Ruby on Rails, ASP.NET, Django, Angular та інших.

Переваги:

- Універсальність: Підтримка різних мов дозволяє використовувати MVC в різних середовищах.
- Широка підтримка: Велика кількість інструментів та бібліотек для роботи з MVC.

Приклад: Використання Angular для Front-end і Django для Back-end зберігає принципи MVC, полегшуючи взаємодію між компонентами.

7. Впровадження сучасних патернів

MVC дозволяє легко впроваджувати сучасні патерни та практики розробки.

Переваги:

- Сумісність з іншими патернами: Легко інтегрується з патернами Dependency Injection, Repository Pattern тощо.
- Зниження складності: Підтримує дотримання принципів SOLID, сприяючи кращій архітектурі програмного забезпечення.

Приклад: Використання патерну "Observer" для автоматичного оновлення View при зміні Model забезпечує сучасний і гнучкий підхід до розробки.

MVC є актуальною архітектурною схемою, яка забезпечує чітке розділення обов'язків, полегшує тестування і підтримку, дозволяє повторне використання компонентів і полегшує адаптацію до змін. Ця архітектура широко використовується в сучасній розробці програмного забезпечення завдяки своїй ефективності, гнучкості і здатності адаптуватися до сучасних вимог і технологій.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І.ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		18

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки веб-додатку було обрано мову програмування JavaScript, а саме – було використано стек MERN(MongoDB, Express.js, React.js, Node.js) для побудови системи.

MERN-стек (MongoDB, Express.js, React.js, Node.js) є сучасною архітектурою для створення веб-додатків. Його актуальність та популярність серед розробників обумовлена рядом факторів, які роблять його привабливим для сучасної веб-розробки.

Що таке MERN-стек?

- MongoDB: Документо-орієнтована база даних NoSQL, яка зберігає дані у форматі JSON-подібних документів.
- Express.js: Легкий і гнучкий веб-фреймворк для Node.js, який полегшує розробку серверної логіки.
- React.js: Фреймворк для розробки користувацького інтерфейсу, створений Facebook, що дозволяє будувати складні інтерфейси за допомогою компонентів.
- Node.js: Серверне середовище для виконання JavaScript, яке дозволяє створювати високопродуктивні і масштабовані додатки.

Актуальність MERN-стеку:

1. Уніфікація технологій на JavaScript

Усі компоненти MERN-стеку використовують JavaScript, що забезпечує єдину мову для клієнтської та серверної частин.

Переваги:

- Зменшення кривої навчання: Розробники можуть зосередитися на одній мові, що полегшує навчання і роботу.
- Швидкий розвиток: Зменшує час на перемикання контексту між мовами.
- Повторне використання коду: Можливість використовувати один і той самий код на клієнті та сервері.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Приклад: Компонент для валідації форм може бути використаний і на сервері, і на клієнті.

2. Гнучкість і модульність

Кожен компонент MERN-стеку легко налаштовується і масштабується, що дозволяє адаптувати додаток під специфічні вимоги проекту.

Переваги:

- Компонентність React: Дозволяє розробляти повторно використовувані інтерфейси.
- Гнучка серверна логіка: Express.js легко інтегрується з іншими середовищами і сервісами.
- Масштабована база даних: MongoDB дозволяє легко масштабуватися і працювати з великими обсягами даних.

Приклад: Додаток для електронної комерції може легко адаптуватися для різних категорій товарів і способів оплати.

3. Висока продуктивність і асинхронність

Node.js забезпечує асинхронну модель виконання, яка дозволяє обробляти велику кількість одночасних запитів без блокування.

Переваги:

- Швидкодія: Зменшення часу на обробку запитів і підвищення продуктивності додатка.
- Асинхронні операції: Підтримка асинхронних операцій забезпечує швидку і ефективну обробку I/O запитів.

Приклад: Чат-додатки і реальні-time сервіси можуть обробляти тисячі одночасних з'єднань.

4. Широка екосистема і підтримка

Усі компоненти MERN-стеку мають широку екосистему з великою кількістю інструментів, бібліотек і фреймворків.

Переваги:

- Багато ресурсів: Легкість у знаходженні документації, прикладів і навчальних матеріалів.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				20
Змн.	Арк.	№ докум.	Підпис	Дата		

- Спільнота: Велика кількість розробників та активна підтримка спільноти.

Приклад: Бібліотеки для управління станом у React або middleware для Express.js допомагають прискорити розробку.

5. Повний стек розробки

MERN забезпечує повний стек для розробки додатків, охоплюючи всі аспекти від бази даних до користувацького інтерфейсу.

Переваги:

- Універсальність: Можливість створення як простих, так і складних додатків з використанням одного стека.

- Спрощення інфраструктури: Зменшує складність інфраструктури, оскільки всі компоненти працюють разом ефективно.

Приклад: Розробка системи управління контентом з використанням MongoDB для зберігання даних, Express.js для серверної логіки, React для інтерфейсу та Node.js для забезпечення продуктивності.

6. Легка інтеграція з іншими сервісами

MERN легко інтегрується з іншими веб-сервісами і API, забезпечуючи масштабованість і розширюваність.

Переваги:

- Мікросервіси: Підтримка архітектури мікросервісів для підвищення масштабованості.

- API: Легка інтеграція з RESTful та GraphQL API для розширення функціональності додатка.

Приклад: Інтеграція з платіжними шлюзами або сторонніми сервісами для додаткової функціональності.

7. Підтримка DevOps і CI/CD

MERN-стек підтримує сучасні підходи до розгортання і безперервної інтеграції.

Переваги:

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		21

- Автоматизація: Підтримка інструментів для автоматизації тестування, побудови і розгортання.

- Швидке розгортання: Підтримка безперервного розгортання і оновлення додатків.

Приклад: Використання Jenkins або GitHub Actions для автоматизації процесів розробки та розгортання додатка.

8. Актуальність для сучасних тенденцій веб-розробки

MERN відповідає сучасним вимогам і тенденціям у розробці веб-додатків, таких як SPA, PWA та реальні-time додатки.

Переваги:

- Підтримка SPA і PWA: Легко створювати односторінкові і прогресивні веб-додатки з сучасним функціоналом.

- Реальні-time можливості: Підтримка веб-сокетів і інших технологій для реальні-time функціоналу.

Приклад: Розробка додатків, що дозволяють працювати офлайн і синхронізуються, коли підключення відновлено.

Переваги та недоліки MERN-стека

Переваги:

1. Єдиний стек для клієнтської та серверної частин, базований на JavaScript, що спрощує розробку і знижує бар'єри входу.

2. Можливість створення високопродуктивних та масштабованих додатків за рахунок асинхронної природи Node.js.

3. Широка спільнота та багата екосистема інструментів та бібліотек, що підтримують швидкий розвиток і розширення додатків.

4. Гнучкість та модульність, що дозволяє легко адаптувати додаток до нових вимог і швидко вносити зміни.

5. Підтримка сучасних тенденцій у веб-розробці, таких як створення SPA, PWA, та реальні-time додатків.

Недоліки:

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		22

1. Потребує гарного розуміння JavaScript та асинхронних програмних моделей, що може бути викликом для новачків.
2. Залежність від інструментів і бібліотек, які можуть швидко застарівати або змінюватися, що вимагає постійного оновлення знань та навичок.
3. Складність інтеграції з деякими існуючими системами, які можуть не підтримувати JavaScript.

MERN-стек є актуальним завдяки уніфікації розробки на JavaScript, гнучкості, високій продуктивності і широкій підтримці спільноти. Він добре підходить для створення сучасних веб-додатків, тому саме його і було використано.

Висновки до першого розділу

У ході виконання першого розділу було описано завдання на кваліфікаційну роботу, яке полягає в створенні програмного веб-додатку, з передбаченим функціоналу авторизації, автентифікації, та онлайн замовлення пального на частині клієнта, або його продаж на частині продавця.

Було розглянуто варіанти можливої архітектури для веб-додатку, проаналізовано їх переваги та недоліки, обгрунтовано вибір сучасної архітектури MVC.

MVC є актуальною архітектурною схемою, яка забезпечує чітке розділення обов'язків, полегшує тестування і підтримку, дозволяє повторне використання компонентів і полегшує адаптацію до змін. Ця архітектура широко використовується в сучасній розробці програмного забезпечення завдяки своїй ефективності, гнучкості і здатності адаптуватися до сучасних вимог і технологій.

Проведено аналіз інструментарію та обрано що буде використано в ході розробки інформаційної системи. Як середовище розробки було обрано WebStorm. Серверна та клієнтська частина програмного комплексу буде написана на мові JavaScript, з використанням стеку MERN (MongoDB, Express.js, React.js, Node.js) для побудови системи.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Варіанти використання системи

Додаток для доставки палива на автозаправні станції — це всеосяжний додаток, розроблений з використанням стека MERN (MongoDB, Express.js, React.js, Node.js). Він призначений для спрощення та оптимізації процесу доставки палива для власників автозаправних станцій і клієнтів.

Особливості варіантів використання системи

Реєстрація та автентифікація користувачів: Користувачі можуть створювати облікові записи, безпечно входити в систему та керувати інформацією свого профілю.

Управління автозаправними станціями: Власники автозаправних станцій можуть реєструвати свої станції, оновлювати наявність палива, ціни та керувати графіками доставки.

Замовлення палива: Клієнти можуть переглядати зареєстровані автозаправні станції, дивитися наявність палива, ціни та робити замовлення на доставку палива.

Інтеграція з безпечними платіжними системами: Інтеграція з безпечними платіжними шлюзами дозволяє клієнтам здійснювати онлайн-платежі за їхні замовлення на паливо.

Відстеження замовлень в режимі реального часу: Клієнти можуть відстежувати статус своїх замовлень на паливо в режимі реального часу — від підтвердження до відправки та доставки.

Сповіщення: Автоматичні сповіщення та оновлення інформують клієнтів про статус замовлення, графіки доставки та будь-які зміни.

Рейтинги та відгуки: Користувачі можуть залишати відгуки та оцінки свого досвіду доставки палива, допомагаючи власникам автозаправних станцій покращувати свої послуги.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		24

2.2 Розробка бази даних системи

Після проведеного аналізу було визначено, що найкращим вибором для зберігання даних є нереляційна система управління базами даних MongoDB. Вибір було зроблено після аналізу декількох ключових факторів.

MongoDB — це документо-орієнтована база даних NoSQL, яка використовується багатьма сучасними веб-додатками, зокрема для онлайн-продажу палива. Її актуальність і переваги в контексті таких додатків включають гнучкість роботи з даними, високу продуктивність і здатність масштабуватися.

Актуальність MongoDB у веб-додатках для онлайн-продажу палива

1. Гнучка структура даних

MongoDB використовує JSON-подібні документи, що дозволяє зберігати дані в динамічній, неструктурованій формі.

Переваги:

- Гнучкість у зберіганні даних: Легкість в управлінні різнорідними даними (наприклад, інформацією про різні види палива з різними атрибутами).
- Легке оновлення структури: Додаток може еволюціонувати без складних міграцій бази даних.

Приклад: Зберігання інформації про пально-мастильні матеріали з різними характеристиками (наприклад, вид, октанове число, виробник) в одному документі.

2. Висока продуктивність і масштабованість

MongoDB підтримує горизонтальне масштабування через шардинг і має вбудовані механізми для індексації та кешування.

Переваги:

- Підтримка великих обсягів даних: Зберігання і обробка великих наборів даних, таких як історія замовлень чи дані про ціни.
- Швидкий доступ: Використання індексів для прискорення запитів, таких як пошук по категоріям палива.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Приклад: Використання шардінгу для розподілу даних по серверах, що забезпечує швидкий доступ до інформації навіть при високому навантаженні.

3. Підтримка асинхронних операцій і реплікації

MongoDB забезпечує асинхронну реплікацію даних і автоматичне відновлення у разі збоїв.

Переваги:

- Безперервна доступність: Мінімізація простоїв за рахунок автоматичної реплікації і відновлення даних.

- Висока доступність: Можливість читання даних з вторинних реплік для зниження навантаження на основний вузол.

Приклад: Застосування реплікації для зберігання даних у кількох дата-центрах, що забезпечує доступність сервісу навіть при збої одного з центрів.

4. Потужні інструменти для агрегації даних

MongoDB має вбудовані механізми для агрегації, що дозволяє легко отримувати складні аналітичні дані.

Переваги:

- Аналіз даних: Можливість виконання складних запитів для аналізу продажів, визначення популярних продуктів та трендів.

- Гнучкі звіти: Генерація звітів без необхідності в складному SQL.

Приклад: Аналіз історії покупок користувачів для визначення популярних видів палива або періодів підвищеного попиту.

5. Підтримка транзакцій

MongoDB підтримує ACID-транзакції на рівні документів і колекцій.

Переваги:

- Цілісність даних: Забезпечення консистентності даних у разі одночасного оновлення.

- Безпека операцій: Виконання складних операцій, таких як оформлення замовлення, як єдине ціле.

Приклад: Забезпечення цілісності даних при оформленні замовлення палива, включаючи оновлення наявності і запис деталей замовлення.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		26

6. Висока доступність і стійкість до відмов

MongoDB підтримує автоматичне перемикавання на резервний вузол у разі збою основного.

Переваги:

- Надійність: Захист від втрати даних і забезпечення безперебійної роботи.
- Безперервна робота: Автоматичне відновлення роботи без втручання адміністратора.

Приклад: При збоях на основному сервері система автоматично переключиться на резервний, забезпечуючи безперебійний доступ до додатка.

7. Швидка інтеграція та гнучкість розвитку

MongoDB легко інтегрується з іншими технологіями та підтримує різноманітні мови програмування.

Переваги:

- Швидкість розробки: Легке інтегрування з веб-фреймворками і бібліотеками.
- Гнучкість: Можливість швидко адаптуватися до змін у вимогах проекту.

Приклад: Використання MERN-стеку (MongoDB, Express.js, React.js, Node.js) для швидкого розробки і розширення функціональності додатка.

8. Глобальне розповсюдження і мульти-регіональні реплікації

MongoDB підтримує реплікацію даних у кількох регіонах.

Переваги:

- Швидкий доступ: Зменшення затримок для користувачів у різних регіонах.
- Місцеве відповідність: Забезпечення відповідності локальним регуляторним вимогам щодо зберігання даних.

Приклад: Розподілення даних у різних географічних зонах для прискорення доступу користувачів і дотримання вимог GDPR.

9. Підтримка гнучких моделей даних і оновлень

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		27

MongoDB дозволяє динамічно оновлювати структуру документів без необхідності складних міграцій.

Переваги:

- Швидкі зміни: Легке додавання нових полів і моделей.
- Менше складнощів: Відсутність необхідності перетворення всієї бази даних для підтримки нових вимог.

Приклад: Додавання нових властивостей до моделей пального без потреби в оновленні всієї структури бази даних.

MongoDB є потужним інструментом для створення веб-додатків, зокрема для онлайн-продажу пального. Її гнучкість у зберіганні та обробці даних, висока продуктивність і здатність масштабуватися роблять її особливо підходящою для цієї задачі. Завдяки підтримці асинхронних операцій, інструментів агрегації, транзакцій, реплікації і інтеграції з іншими сервісами, MongoDB дозволяє створювати надійні, ефективні і гнучкі рішення для сучасних бізнес-завдань.

На основі вимог до системи було розроблено концептуальну модель бази даних, яка включає кілька основних таблиць для зберігання даних про клієнтів, продавців та замовлення

Так в нашій базі даних виглядає один запис з таблиці orders (замовлення):

```
{
  "_id": {
    "$oid": "667844ca74d8ccc4de9b381c"
  },
  "address": {
    "lat": -47.04018214480665,
    "lng": 53.156243562698364
  },
  "fuel": {
    "petrol": {
      "price": 11,
```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "quantity": 11
  },
  "diesel": {
    "price": 1,
    "quantity": 1
  }
},
"method": {
  "cash": 332562
},
"userId": {
  "$oid": "66783ef9772c7eed72ca2b29"
},
"stationId": {
  "$oid": "6678444474d8ccc4de9b3801"
},
"isCanceled": {
  "status": false
},
"isAccepted": {
  "status": true
},
"isDelivered": {
  "status": true
},
"__v": 0
}.

```

Так в MongoDB зберігаються дані про клієнтів веб-додатку:

```

{
  "_id": {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				29
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "$oid": "6679a4e1eae101f63a16d60"
  },
  "name": "Дмитро Д",
  "email": "user@gmail.com",
  "password":
"$2a$12$8RNRJXMKRmOqM6TIue0XNOLOtSfO/oQLcwucJW6iJe8v4DtBdsYdu"
,
  "phone": 3801234567,
  "isValid": true,
  "isBlocked": false,
  "__v": 0
}

```

Так в MongoDB зберігаються дані про продавців-власників АЗС, які зареєструвалися у веб-додатку:

```

{
  "_id": {
    "$oid": "6679a570eae101f63a16d71"
  },
  "name": "okko",
  "owner": "okko",
  "email": "okko@gmail.com",
  "password":
"$2a$12$VzAfji7HZ2x6s8i3GXy6k.mrHorLX6ZzwUl/Hn1ngOjxRRYzATBgG",
  "location": {
    "lat": 50.279178144857845,
    "lng": 28.63807697925645,
    "_id": {
      "$oid": "6679a570eae101f63a16d72"
    }
  },
}

```

		Д.ОДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"phone": 3801234567,
"isValid": true,
"isBlocked": false,
"quantity": {
  "petrol": {
    "price": 59,
    "quantity": 14850
  },
  "diesel": {
    "price": 65,
    "quantity": 65
  }
},
"__v": 0
}

```

2.3 Проектування архітектури інформаційної системи

Для розробки веб-додатку було обрано архітектурний шаблон MVC. Для роботи серверної частини(Backend) системи було реалізовано структуру, яка зображена на рисунку 2.1, а для клієнтської частини(Frontend) було реалізовано структуру, яка зображена на рисунку 2.2.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		31

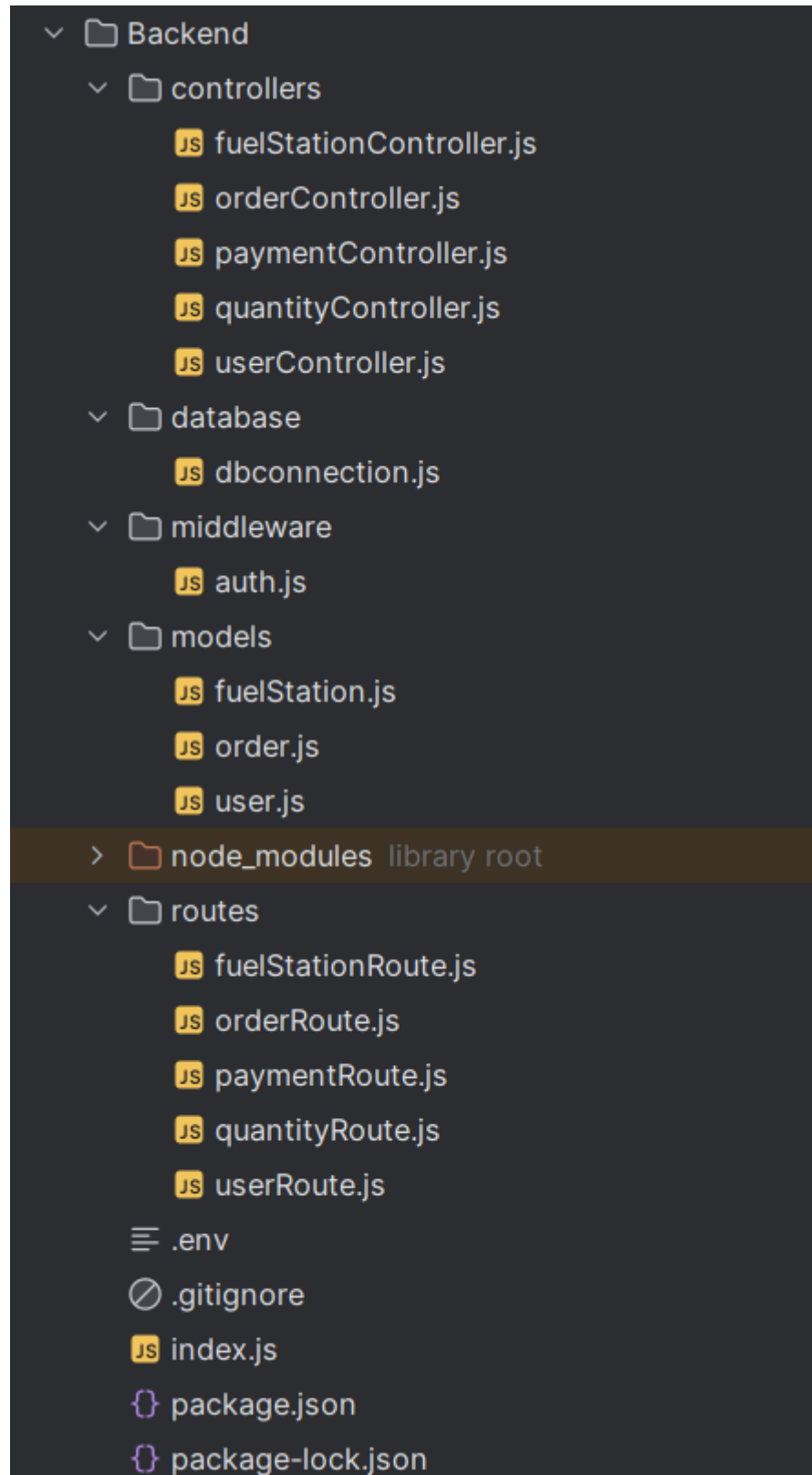


Рисунок 2.1 – Структура серверної частини веб-додатку

Були створені папки controllers, database, middleware, models, routes, в кожній з яких знаходяться файли, які відповідають за свою частину функціоналу веб-додатка.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				32
Змн.	Арк.	№ докум.	Підпис	Дата		

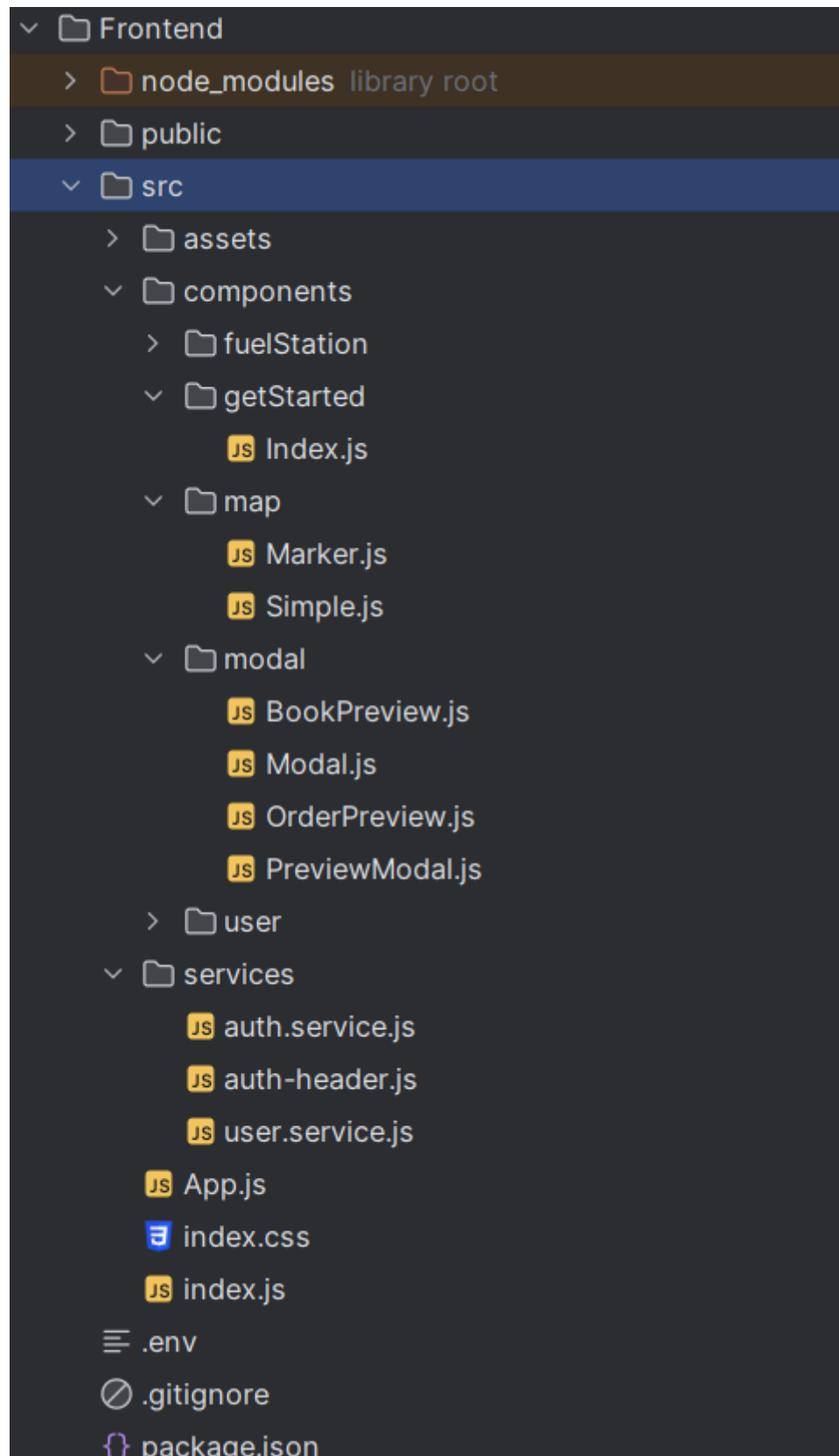


Рисунок 2.2 – Структура серверної частини веб-додатку

На клієнтській частині додатку були створені папки components, services, assets, в кожній з яких знаходяться файли, які відповідають за свою частину функціоналу веб-додатка.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Розглянемо основні класи системи.

Папка

Папка `controllers` у Node.js проєкті містить модулі, які виконують логіку обробки HTTP-запитів, управління бізнес-логікою, взаємодію з моделями, та обробку помилок. Використання контролерів забезпечує добре організовану архітектуру, яка полегшує розробку, тестування та підтримку додатка.

Основні функції папки «controllers»

1. Обробка HTTP-запитів

Контролери відповідають за обробку вхідних HTTP-запитів, які надходять від клієнтів. Вони приймають запити, обробляють дані і формують відповіді, що відправляються назад клієнтам.

Контролери містять основну бізнес-логіку додатка. Вони виконують різні обчислення, перевірки та процеси, необхідні для виконання запитів.

Контролери взаємодіють з моделями для отримання або зміни даних у базі даних. Вони роблять запити до моделей і обробляють відповіді для надання кінцевих даних клієнту.

Контролери також відповідають за обробку помилок, які можуть виникнути під час обробки запитів або взаємодії з базою даних.

Переваги використання контролерів

1. Організація коду

Контролери дозволяють структурувати код за функціональними групами, що покращує його зрозумілість і підтримку.

2. Полегшене тестування

Завдяки чітко визначеним функціям контролерів, тестування різних частин додатка стає простішим.

3. Розподіл відповідальності

Контролери забезпечують чіткий розподіл відповідальності між обробкою запитів, бізнес-логікою і взаємодією з базою даних.

4. Розширюваність

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		34

При додаванні нових функцій або розширенні існуючих, чітка структура контролерів полегшує внесення змін.

Папка `models` у `Node.js` додатку відіграє ключову роль в управлінні даними. Вона містить схеми для структурування даних, методи для CRUD-операцій, логіку валідації та обробки даних, а також визначає відносини між даними. Використання моделей забезпечує чітку, організовану і безпечну роботу з базою даних, спрощує підтримку і розвиток додатка, та дозволяє ефективно реалізовувати бізнес-логіку.

Основні функції папки `'models'`

1. Опис структури даних

Моделі визначають схеми, за якими дані зберігаються в базі даних. Це включає визначення полів, їх типів та обмежень.

2. Взаємодія з базою даних

Моделі використовуються для створення, читання, оновлення і видалення (CRUD) даних у базі. Вони забезпечують абстракцію над запитами до бази даних.

3. Валідація даних

Моделі включають логіку для валідації даних перед їх збереженням у базі даних. Це може включати перевірку типів, обов'язкових полів, унікальності та інші правила.

4. Обробка та нормалізація даних

Моделі можуть містити методи, які обробляють дані перед їх збереженням або після їх отримання з бази даних. Це може включати нормалізацію, шифрування, та інші операції.

5. Визначення відносин між даними

Моделі можуть визначати відносини між різними типами даних, такими як зв'язки "один до багатьох" або "багато до багатьох".

6. Запити з агрегацією

Моделі можуть бути використані для виконання складних запитів, таких як агрегації, які об'єднують дані з кількох документів.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Переваги використання моделей

1. Ізоляція логіки даних

Моделі відокремлюють логіку управління даними від решти коду, спрощуючи обслуговування і розвиток додатка.

2. Централізація обробки даних

Вся логіка, що стосується даних, зібрана в одному місці, що зменшує дублювання коду і спрощує оновлення.

3. Покращена безпека

Валідація і обробка даних в моделях покращує безпеку, запобігаючи введенню некоректних або шкідливих даних у базу.

4. Підтримка відносин і агрегацій

Моделі дозволяють легко визначати складні запити і відносини між даними, що спрощує реалізацію складної бізнес-логіки.

Папка ``routes`` у Node.js додатку виконує важливу роль у визначенні маршрутизації запитів, розподілі обробки запитів між відповідними контролерами, обробці параметрів і використанні `middleware`. Вона допомагає створити організовану структуру для API, спрощує управління маршрутами і покращує розширюваність додатка.

Основні функції папки ``routes``

1. Визначення маршрутів

Файли в папці ``routes`` визначають маршрути або шляхи, які можуть бути викликані за допомогою HTTP-запитів. Вони вказують, який метод контролера слід викликати для кожного маршруту.

2. Призначення методів HTTP

Маршрути визначають, які методи HTTP (GET, POST, PUT, DELETE і т.д.) будуть використовуватися для різних дій в додатку. Це допомагає організувати обробку запитів в залежності від типу операції.

3. Обробка параметрів

Маршрути можуть містити параметри URL, такі як ``:id``, які передаються в функції контролерів для отримання або маніпуляції конкретними ресурсами.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		36

4. Middleware

Маршрути можуть використовувати middleware для обробки запитів до їх досягнення контролера. Middleware може бути використано для автентифікації, авторизації, обробки помилок або іншої попередньої обробки.

5. Організація API

Маршрути дозволяють структурувати API в ієрархічну систему, що робить його більш зрозумілим і організованим. Наприклад, можна групувати маршрути за функціональністю.

6. Логіка помилок

Маршрути можуть бути конфігуровані для обробки специфічних помилок, таких як маршрути, які не знайдені, або обробка специфічних помилок, що виникають при запитах.

Переваги використання папки `routes`

1. Чітка організація

Папка `routes` надає чітку структуру для організації кінцевих точок API, що спрощує підтримку і розвиток додатка.

2. Спрощення обслуговування

Маршрути зосереджують логіку маршрутизації в одному місці, спрощуючи її управління і модифікацію.

3. Гнучкість

Можливість легко додавати нові маршрути або змінювати існуючі маршрути без необхідності значних змін в іншій частині коду.

4. Покращена чіткість

Зрозуміло, як запити обробляються, який контролер відповідає за обробку і які дії виконуються на кожному маршруті.

Папка `components` у React додатку служить для організації, зберігання та управління компонентами інтерфейсу користувача. Вона спрощує створення і підтримку інтерфейсу, забезпечуючи повторне використання компонентів, чіткість структури і можливість модульної розробки.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Основні функції папки `components` в React:

1. Організація компонентів

- Структура: Папка `components` містить підкаталоги або файли, які групують компоненти за їхнім призначенням або функціоналом.
- Модульність: Компоненти розділяються на окремі файли для полегшення керування та зберігання.

2. Повторне використання компонентів

- Композиція: Кожен компонент може бути повторно використаний в різних частинах додатка, що забезпечує ефективне використання коду і прискорює розробку.
- Спрощення обслуговування: Зменшує дублювання коду і полегшує його модифікацію.

3. Структура UI

- Компонентна архітектура: Використання компонентів дозволяє створювати ієрархію UI з різними рівнями складності.
- Чіткість: Папка `components` відображає структуру і логіку, за якою побудований інтерфейс користувача.

4. Підтримка стилів і логіки

- Компонентне стилізування: Компоненти можуть включати не тільки розмітку JSX, а й CSS або інші стилі, що відповідають за їхній зовнішній вигляд.
- Логіка компонентів: Деякі компоненти можуть містити також логіку, яка відповідає за обробку подій або управління станом.

2.4 Реалізація основного функціоналу системи

Внаслідок ретельного аналізу конкурентів було поставлено такі задачі стосовно основного функціоналу системи:

Користувачі можуть створювати облікові записи, безпечно входити в систему та керувати інформацією свого профілю. Можемо побачити як це реалізовано в контролері клієнта:

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

import User from "../models/user.js";
import jwt from "jsonwebtoken";
import bcrypt from "bcryptjs";

export const signup=async(req,res)=>{
  const {name,email,password,phone}=req.body;
  try {
    const user=await User.findOne({email});
    if(user){
      return res.status(301).json({message:"User already exists with that email"})
    }

    const hashedPassword=await bcrypt.hash(password,12);

    const newUser=await User.create({
      name,
      email,
      password:hashedPassword,
      phone,

    })
    await newUser.save();
    if(newUser){
      return res.status(201).json({message:"User created successfully"});
    }

  } catch (error) {
    res.status(301).json(error)
  }
}

export const login = async (req, res) => {
  const { email, password } = req.body;
  let user;
  try {
    user = await User.findOne({ email });
    if (!user) {
      return res.status(301).json({ message: "User with that email is not exists" });
    }
    const isValidPassword = await bcrypt.compare(password, user.password);
    if (!isValidPassword) {
      return res.status(300).json({ message: "email and password are incorrect" });
    }
    const authToken = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1hr" });
  }
}

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		39

```

        res.status(200).json({ authToken ,userId:user._id});
    } catch (error) {
        res.status(400).json(error);
    }
}

export const getUserInfo = async (req,res) =>{
    try{
        const user = await User.findOne({_id:req.params.id})
        if(!user){
            return res.status(301).json({ message: "User does not exists" });
        }
        return res.status(201).json(user)
    }catch(error){
        res.status(400).json(error);
    }
}

export const changePassword = async (req,res) =>{
    const { password,newPassword,userId } = req.body;
    try {
        const user = await User.findById(userId);

        if (!user) {
            return res.status(400).json({ message: "User doesn't Exist" })
        }

        const isValidPassword = await bcrypt.compare(password, user.password)
        if (isValidPassword) {
            const hashedPassword = await bcrypt.hash(newPassword, 12);

            const updatedPassword = await User.updateOne({
                _id : userId
            },{
                $set : {
                    password : hashedPassword
                }
            })

            if(!updatedPassword){
                return res.status(400).json({ message: "Updating Failure" })
            }
            return res.status(200).json({ message: "Password Updated Successfully" })
        }else{
            return res.status(400).json({ message: "Current Password not matching" })
        }
    }
}

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (error) {
      res.status(400).json(error)
    }
  }
}

```

Власники автозаправних станцій можуть реєструвати свої станції, оновлювати наявність палива, ціни та керувати графіками доставки. Приклад реалізації даного функціоналу можемо побачити у контролері продавця:

```

import Station from "../models/fuelStation.js";
//import Quantity from "../models/quantity.js";
import jwt from "jsonwebtoken";
import bcrypt from "bcryptjs";

export const registerStation = async (req, res) => {
  const { name, owner, email, password, location, phone } = req.body;
  const { lat, long } = location;
  try {
    const registeredUser = await Station.findOne({
      email
    })
    if (registeredUser) {
      return res.status(400).json({ message: "Already registered" })
    }

    const usedLocation = await Station.findOne({
      "location.lat": lat, "location.long": long
    })

    if (usedLocation) {
      return res.status(400).json({ message: "Location already added" })
    }

    const hashedPassword = await bcrypt.hash(password, 12);
    const newRegisteredUser = await Station.create({
      name,
      owner,
      email,
      password: hashedPassword,
      location,
      phone,
      quantity: {
        petrol: {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        price: 0,
        quantity: 0
      },
      diesel: {
        price: 0,
        quantity: 0
      }
    }
  })

  await newRegisteredUser.save();
  if (newRegisteredUser) {
    res.status(201).json({ message: "Registered successfully" })
  }

} catch (error) {
  res.status(400).json(error)
}
}

export const loginStation = async (req, res) => {
  const { email, password } = req.body;
  let loginUser;
  let token;

  try {
    loginUser = await Station.findOne({ email });

    if (!loginUser) {
      return res.status(301).json({ message: "Station with that email does not exists" })
    }

    const isValidPassword = await bcrypt.compare(password, loginUser.password)
    if (!isValidPassword) {
      return res.status(300).json({ message: "email and password are incorrect" });
    }
    token = jwt.sign({ id: loginUser._id }, process.env.JWT_SECRET);
    res.status(200).json({ token, stationId: loginUser._id })

  } catch (error) {
    res.status(400).json(error)
  }
}

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		42

```

}

export const updateFuel = async (req,res) =>{
  const { quantity,stationId } = req.body;
  try {
    const user = await Station.findById(stationId);

    if (!user) {
      return res.status(400).json({ message: "User doesn't Exist" })
    }
    const updatedUser = await Station.updateOne({
      _id : stationId
    },{
      $set : {
        quantity : quantity
      }
    })

    if(!updatedUser){
      return res.status(400).json({ message: "Updating Failure" })
    }
    return res.status(200).json({ message: "Updated Successfully" })
  } catch (error) {
    res.status(400).json(error)
  }
}

export const changePassword = async (req,res) =>{
  const { password,newPassword,stationId } = req.body;
  try {
    const user = await Station.findById(stationId);

    if (!user) {
      return res.status(400).json({ message: "User doesn't Exist" })
    }

    const isValidPassword = await bcrypt.compare(password, user.password)
    if (isValidPassword) {
      const hashedPassword = await bcrypt.hash(newPassword, 12);

      const updatedPassword = await Station.updateOne({
        _id : stationId
      },{
        $set : {

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арх.
		А.І. ОРІНЧАК				
Змн.	Арх.	№ докум.	Підпис	Дата		43

```

        password : hashedPassword
    }
})

if(!updatedPassword){
    return res.status(400).json({ message: "Updating Failure" })
}
return res.status(200).json({ message: "Password Updated Successfully" })
}else{
    return res.status(400).json({ message: "Current Password not matching" })
}
} catch (error) {
    res.status(400).json(error)
}
}

export const getFuelStations = async (req, res) => {
    let stations;
    try {
        stations=await Station.find({})
        if (!stations) {
            return res.status(400).json({ message: "Stations not found" })
        }

        res.status(200).json({ stations});
    } catch (error) {
        console.log(error,"err")
        res.status(400).json(error)
    }
}

export const getStationWithId = async (req, res) => {
    let station;
    try {
        station = await Station.findById(req.params.id);
        if (!station) {
            return res.status(400).json({ message: "Station with that id does not exists" })
        }
        res.status(200).json(station);
    } catch (error) {
        res.status(400).json(error)
    }
}
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				44
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнти можуть переглядати зареєстровані автозаправні станції, дивитися наявність палива, ціни та робити замовлення на доставку палива. Інтеграція з безпечними платіжними шлюзами дозволяє клієнтам здійснювати онлайн-платежі за їхні замовлення на паливо. Клієнти можуть відстежувати статус своїх замовлень на паливо в режимі реального часу — від підтвердження до відправки та доставки. Автоматичні сповіщення та оновлення інформують клієнтів про статус замовлення, графіки доставки та будь-які зміни. Реалізацію даного функціоналу можна побачити в контролері замовлення в додатках.

Користувачі можуть залишати відгуки та оцінки свого досвіду доставки палива, допомагаючи власникам автозаправних станцій покращувати свої послуги.

Також користувачі мають реєструватися та авторизуватися у веб-додатку за допомогою спеціального jwt-токена, приклад реалізації даної системи:

```
import jwt from "jsonwebtoken";
export const verifyToken = (req, res, next) => {
  const token = req.headers.authorization;
  if (token) {
    jwt.verify(token, process.env.JWT_SECRET, (err) => {
      if (err) return res.sendStatus(403);
      next();
    });
  } else {
    res.sendStatus(401);
  }
}
```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Висновки до другого розділу

В результаті виконання другого розділу кваліфікаційної роботи було визначено та описано варіанти використання інформаційної системи. Система передбачає дві ролі: користувач-клієнт та користувач-продавець. В кожного з користувачів є свої унікальні можливості щодо використання веб-додатку. У процесі проектування веб-додатку було визначено основні бізнес-вимоги додатку та основні частини функціоналу кожного користувача.

Було також розроблено структуру системи. Вона налічує декілька модулів, кожен з яких працює окремо. Спроектовано архітектуру, при розробці системи використано патерн MVC. Розглянуто основні класи серверної та клієнтської частини додатку.

База даних налічує 3 таблиці та містить такі зв'язки як один до одного, один до багатьох, багато до одного, багато до багатьох.

Розроблено та предствалено функціональні алгоритми, розглянуто алгоритми взаємодії користувачів з функціоналом додатку, також розглянутий процес реєстрації та аутентифікації користувача.

Для написання веб-додатку продукту було використано об'єкто-орієнтований підхід з дотриманням його основних принципів, а саме: інкапсуляція, поліморфізм, наслідування.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		46

РОЗДІЛ 3 ОГЛЯД ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Огляд інтерфейсу системи

При вході на головну сторінку веб-додатку користувачу показується форма з вибором його ролі, зображена на рисунку 3.1.

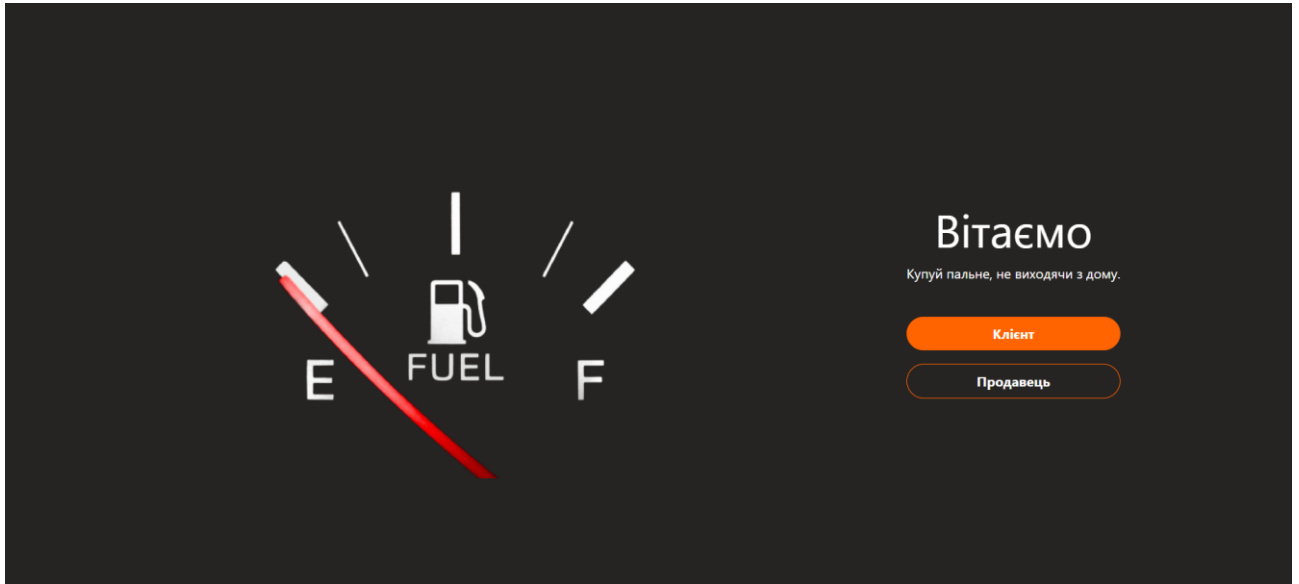


Рисунок 3.1 – Головна сторінка

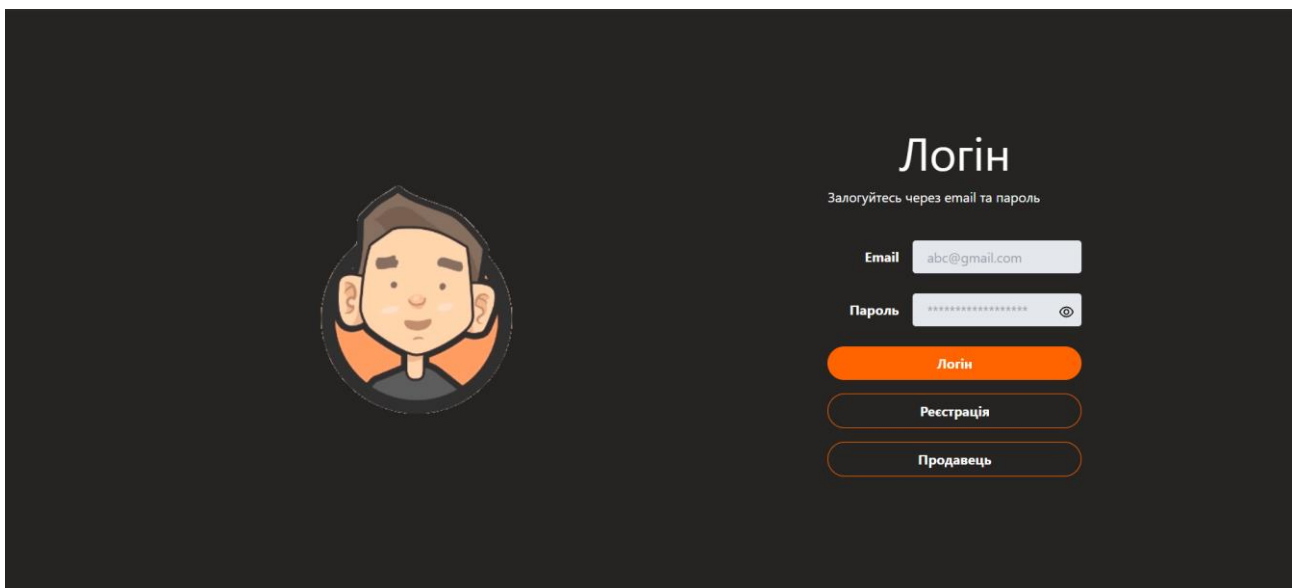


Рисунок 3.2 – Форма логіну користувача

Якщо користувач обрав варіант «Клієнт», то йому відкривається форма логіну користувача, зображена на рисунку 3.2. Якщо клієнт не зареєстрований, то йому видасть помилку, зображену на рисунку 3.3

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		47

The screenshot shows a login interface with the title "Логін" (Login). Below the title is the instruction "Залогуйтесь через email та пароль" (Log in via email and password). There are two input fields: "Email" with the value "user@gmail.com" and "Пароль" (Password) with masked characters. Below the fields are three buttons: "Логін" (Login), "Реєстрація" (Registration), and "Продавець" (Seller). A red error message box at the top right states: "User with that email is not exists".

Рисунок 3.3 – Приклад помилки під час логіну

При переході на форму «Реєстрація» для нових користувачів, користувача відкривається форма реєстрації, зображена на рисунку 3.4.

The screenshot shows a registration interface with the title "Реєстрація" (Registration). Below the title is the instruction "Зареєструйтесь через email та пароль" (Register via email and password). On the left is a hand icon showing a peace sign. On the right are four input fields: "Ім'я" (Name) with the value "ПІБ", "Email" with the value "abc@gmail.com", "Телефон" (Phone) with the value "+380XXXXXXXX", and "Пароль" (Password) with masked characters. Below the fields are two buttons: "Зареєструватись" (Register) and "Логін" (Login).

Рисунок 3.4 – Форма реєстрації користувача

Якщо клієнт успішно увійшов до системи, він побачить головну сторінку зображену на рисунку 3.5. Він може побачити свою геолокацію (якщо він дозволив передачу своєї геопозиції у браузері), а також 4 кнопки: «Замовити пальне», «Історія», «Профіль», та «Вийти».

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		48

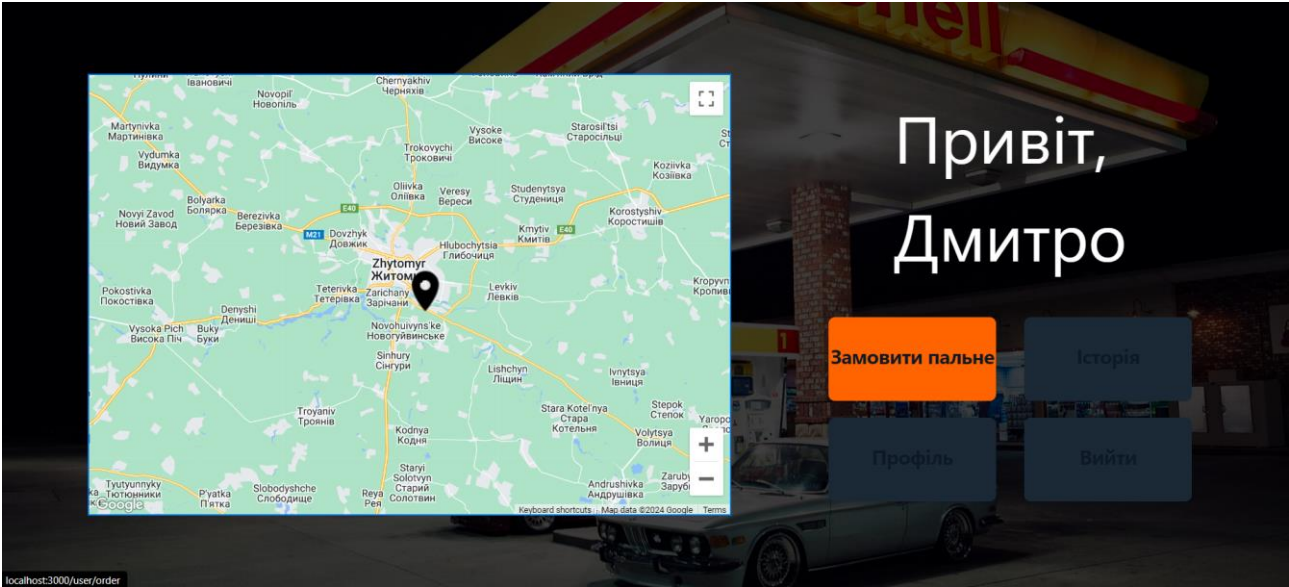


Рисунок 3.5 – Головна сторінка користувача

При натисканні кнопки «Профіль» клієнт зможе змінити свій пароль, використовуючи форму зображену на рисунку 3.6.

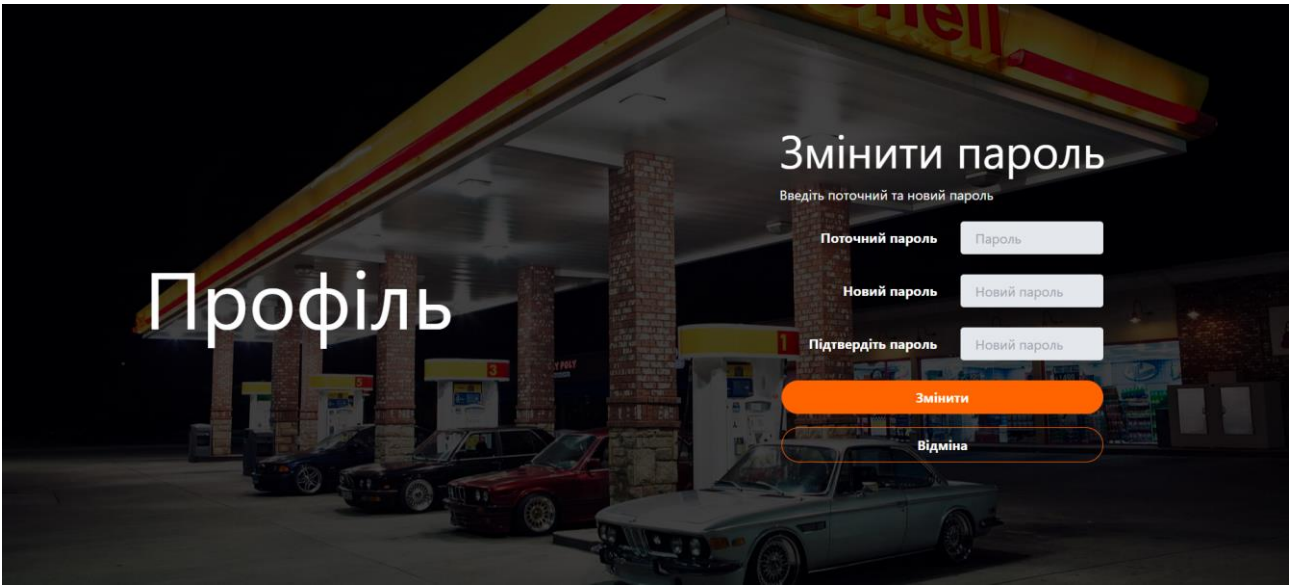


Рисунок 3.6 – Форма зміни паролю користувача

При натисканні кнопки «Замовлення» клієнт зможе обрати бажану станцію зі списку станцій, орієнтуючись по назві, кількості палива, ціни палива, або ж відстані до його місцезнаходження. Даний інтерфейс зображений на рисунку 3.7.

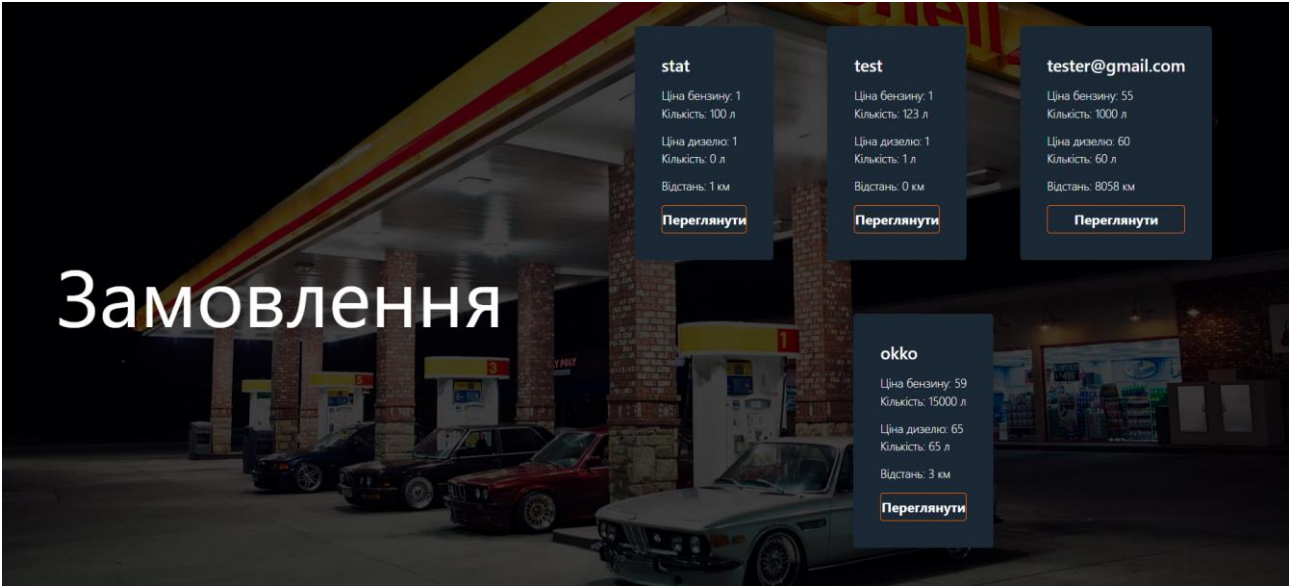


Рисунок 3.7 – Список станцій з паливом, доступних для замовлення

Коли користувач визначиться зі станцією, йому висвітиться інформація з місцезнаходженням станції на карті та цінами і кількістю доступного пального, як це зображено на рисунку 3.8.

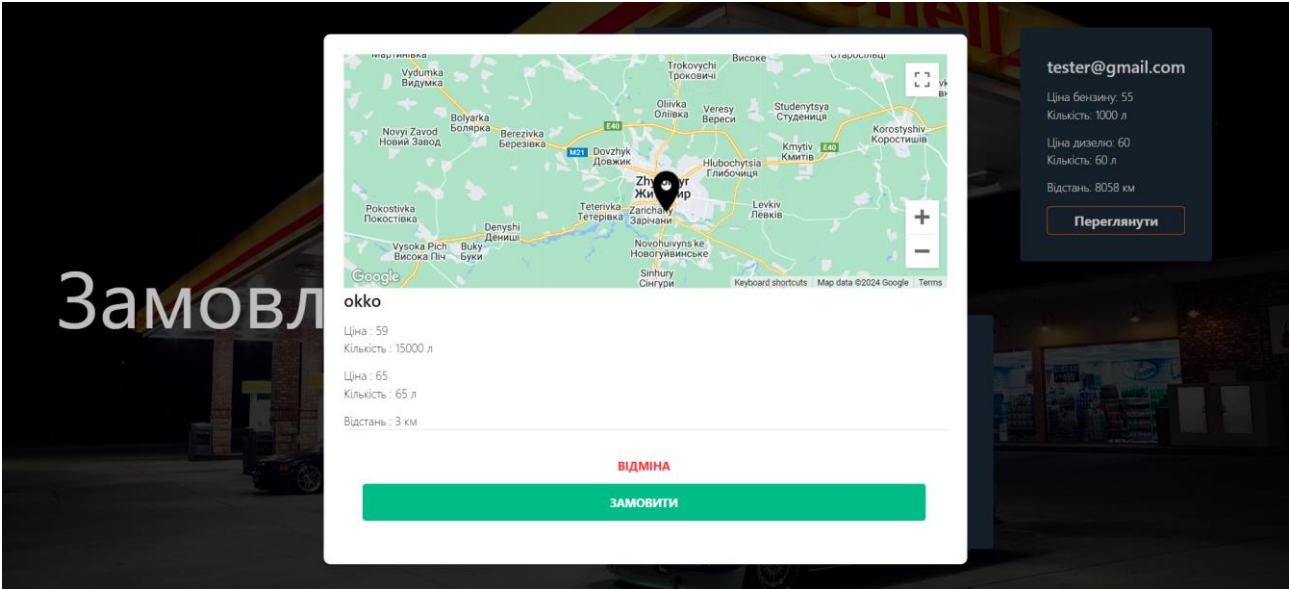


Рисунок 3.8 – Геолокація станції з паливом

При натисканні кнопки «Замовити», клієнту відкриється форма, зображена на рисунку 3.9, за допомогою якої він зможе обрати кількість потрібного пального та точку, куди його можна доставити.

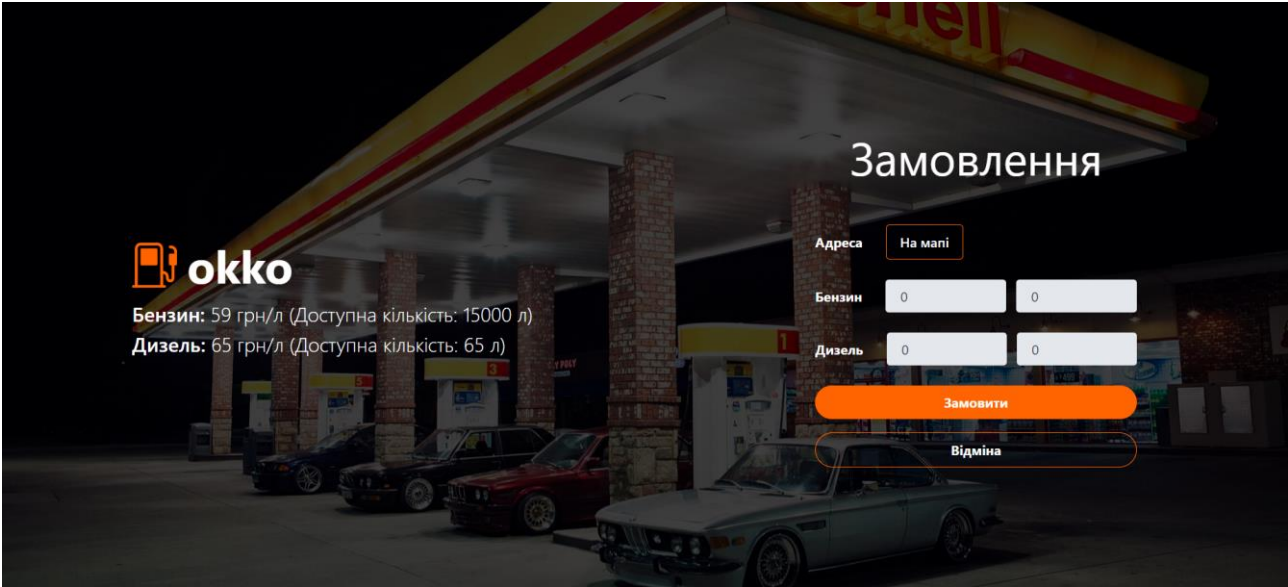


Рисунок 3.9 – Форма оформлення замовлення

Якщо клієнт прийняв рішення оформити замовлення, він матиме змогу перевірити всі дані на модальній формі «Чек», яка зображена на рисунку 3.10, та обрати метод оплати «Онлайн» для розрахування одразу, або ж метод «Готівка», якщо клієнт хоче оплатити під час отримання доставки.

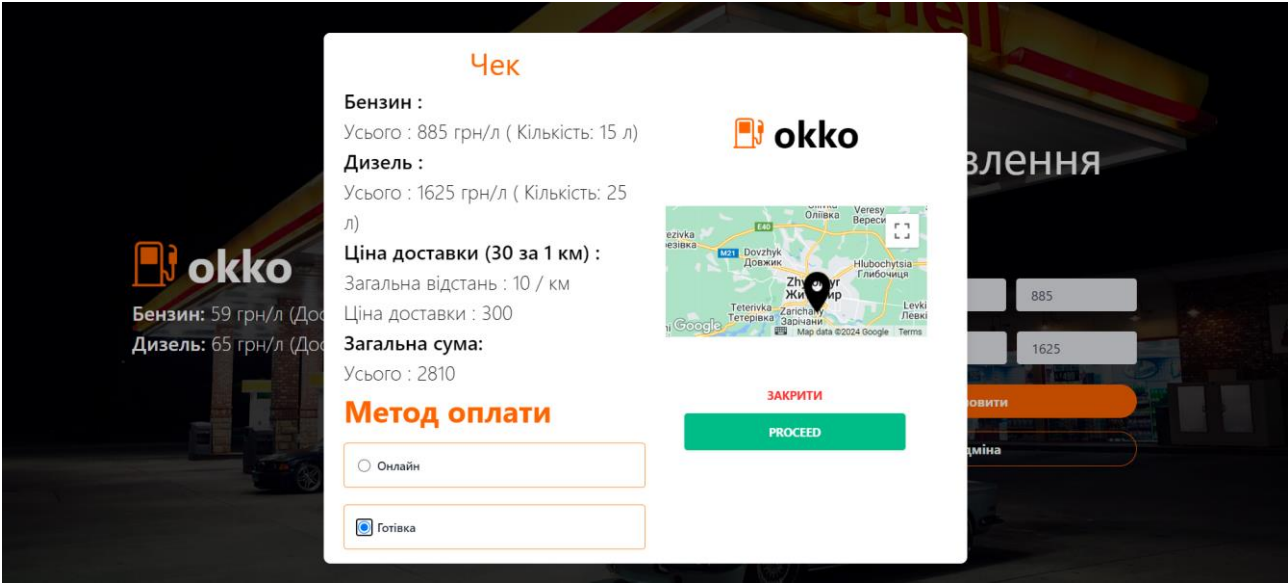


Рисунок 3.10 – Форма підтвердження та оплати замовлення

При натисканні кнопки «Історія», клієнт матиме змогу побачити весь список його замовлень та їх статуси відповідно. Даний інтерфейс зображено на рисунку 3.11.

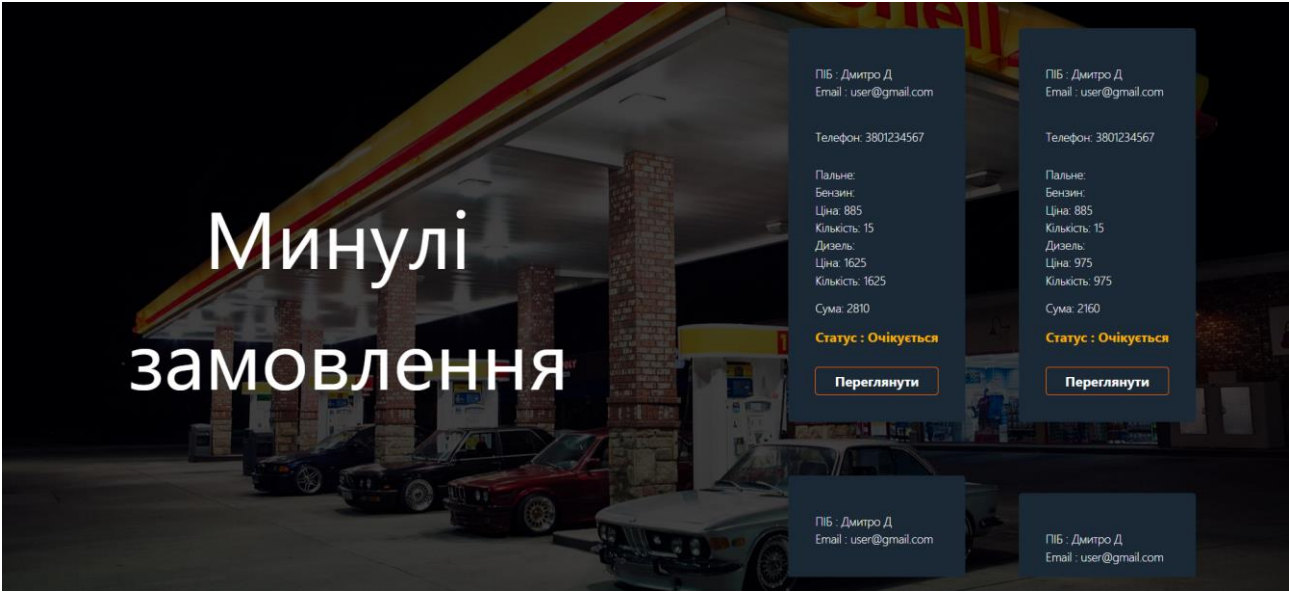


Рисунок 3.11 – Список замовлень користувача

Як зображено на рисунку 3.12, клієнт має змогу перевірити всі деталі замовлення та побачити його на карті, при натисканні кнопки «Переглянути».

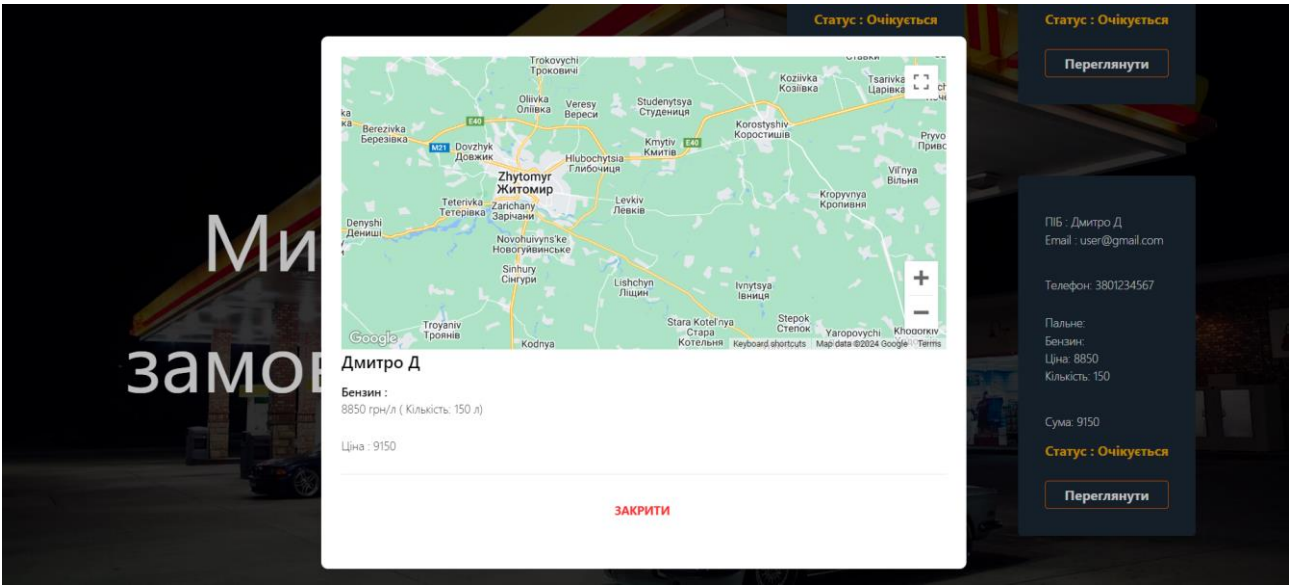


Рисунок 3.12 – Перегляд замовлення користувача

На рисунках 3.13 та 3.14 зображені замовлення клієнта з різними статусами. «Очікується» - клієнт оформив замовлення, очікується підтвердження від постачальника. «В дорозі» - постачальник прийняв замовлення і воно знаходиться в процесі доставки. «Відміна» - постачальник відмовив клієнту в його замовленні. «Доставлено» - постачальник виконав замовлення клієнту та підтвердив його отримання.

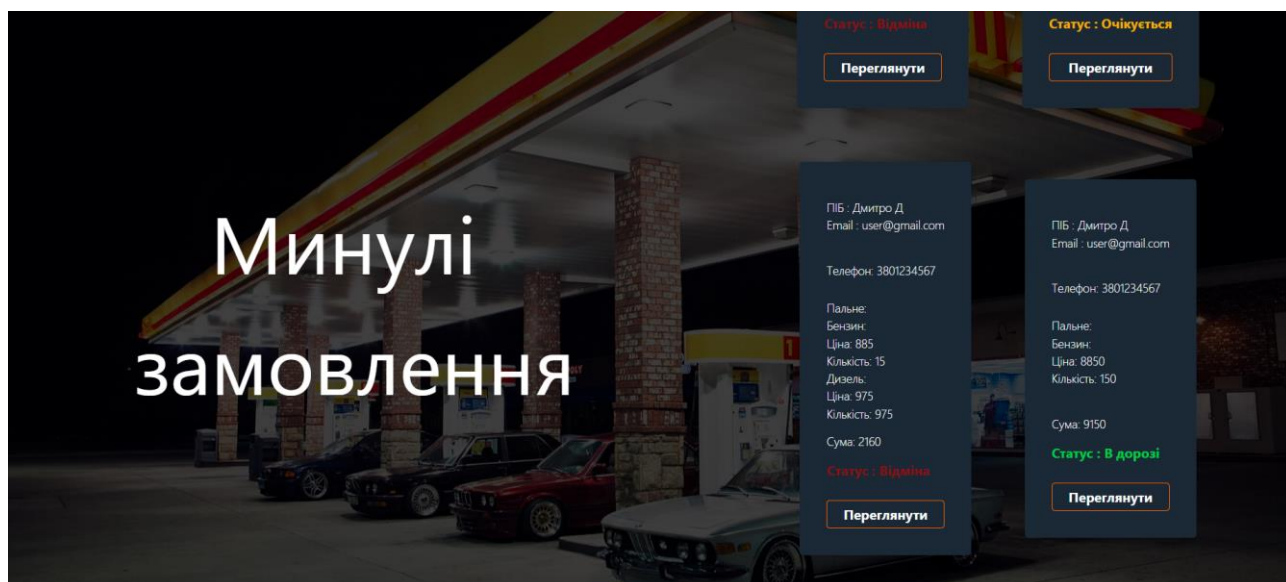


Рисунок 3.13 – Зміна статусу замовлення

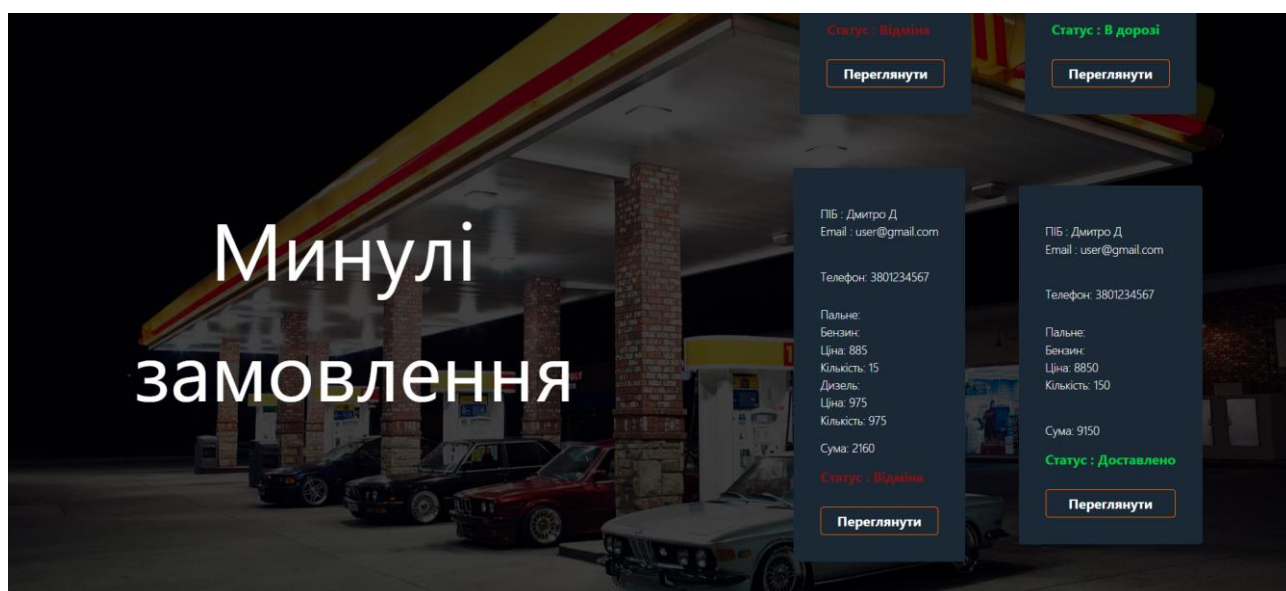


Рисунок 3.14 – Замовлення доставлено

Якщо користувач обрав варіант «Продавець», то йому відкривається форма логіну продавця, зображена на рисунку 3.15.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		53



Логін

Залогуйтесь через email та пароль

Email

Пароль

[Логін](#)

[Зареєструватись](#)

[Користувач](#)

Рисунок 3.15 – Форма логіну постачальника

При переході на форму «Реєстрація» для нових продавців, у користувача відкривається форма реєстрації продавця, зображена на рисунку 3.16.



Реєстрація

Зареєструйтесь через email та пароль

Назва станції

ПІБ власника

Email

Телефон

Пароль

Локація [Відкрити карту](#)

[Зареєструватись](#)

[Логін](#)

Рисунок 3.16 – Форма реєстрації користувача

При реєстрації продавця він має змогу вказати своє точне місцезнаходження на карті, що зображено на рисунку 3.17.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		54

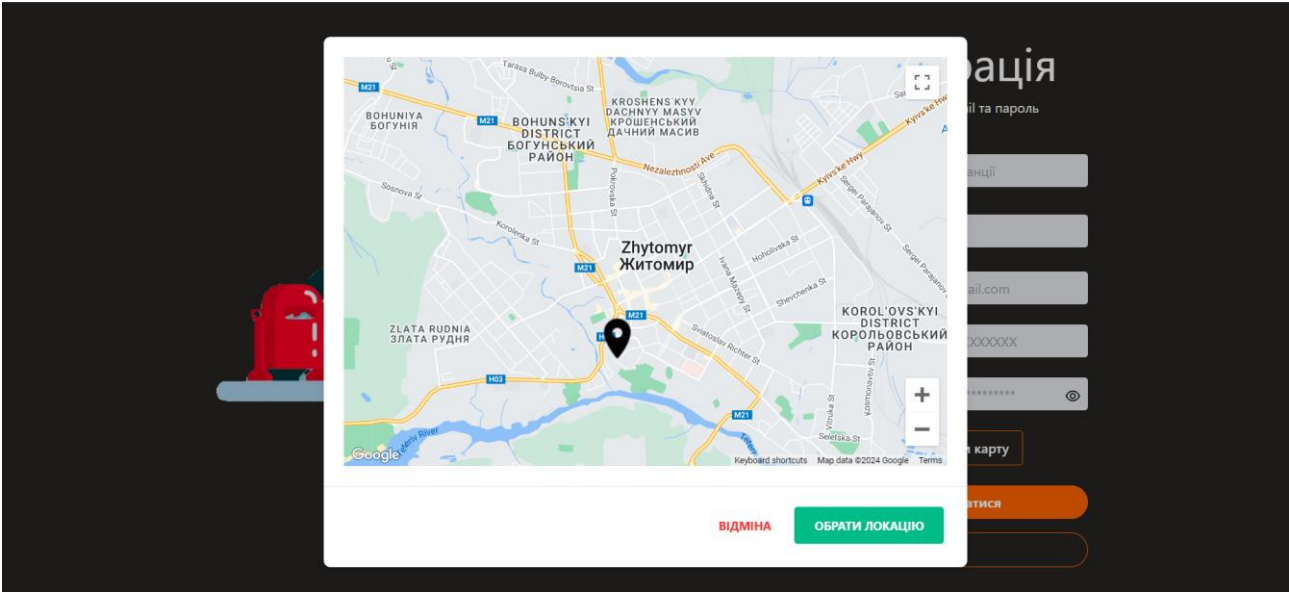


Рисунок 3.17 – Вибір геолокації постачальника

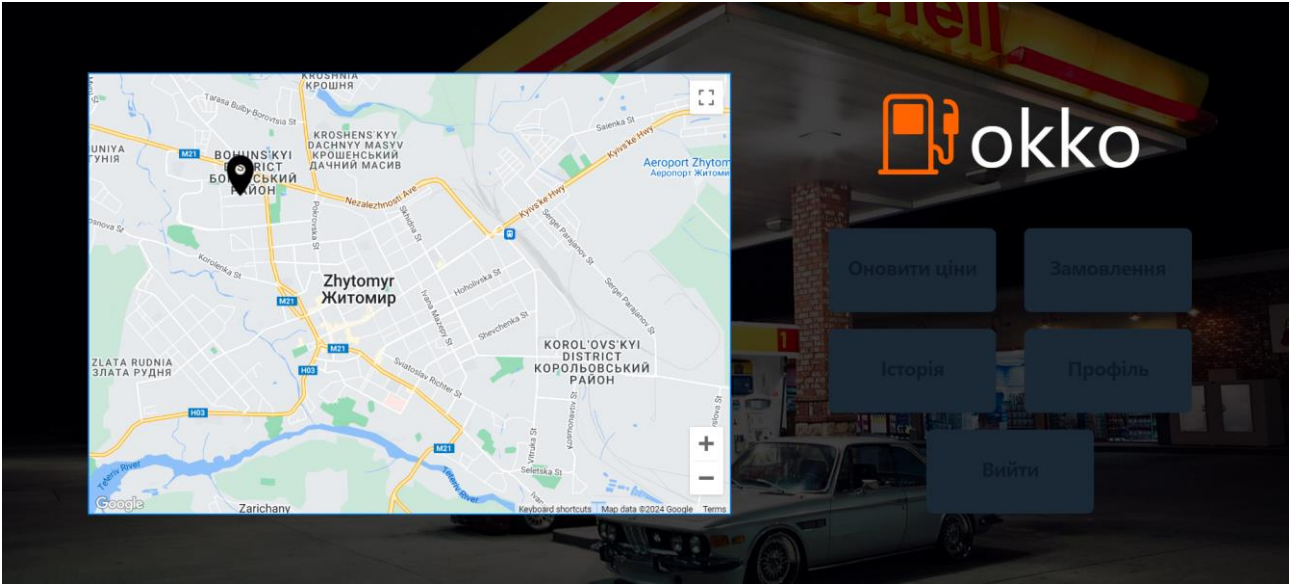


Рисунок 3.18 – Головна сторінка постачальника

Якщо продавець успішно увійшов до системи, він побачить головну сторінку зображену на рисунку 3.18. Він може побачити свою геолокацію, а також 5 кнопок: «Оновити ціни», «Замовлення», «Історія», «Профіль», та «Вийти».

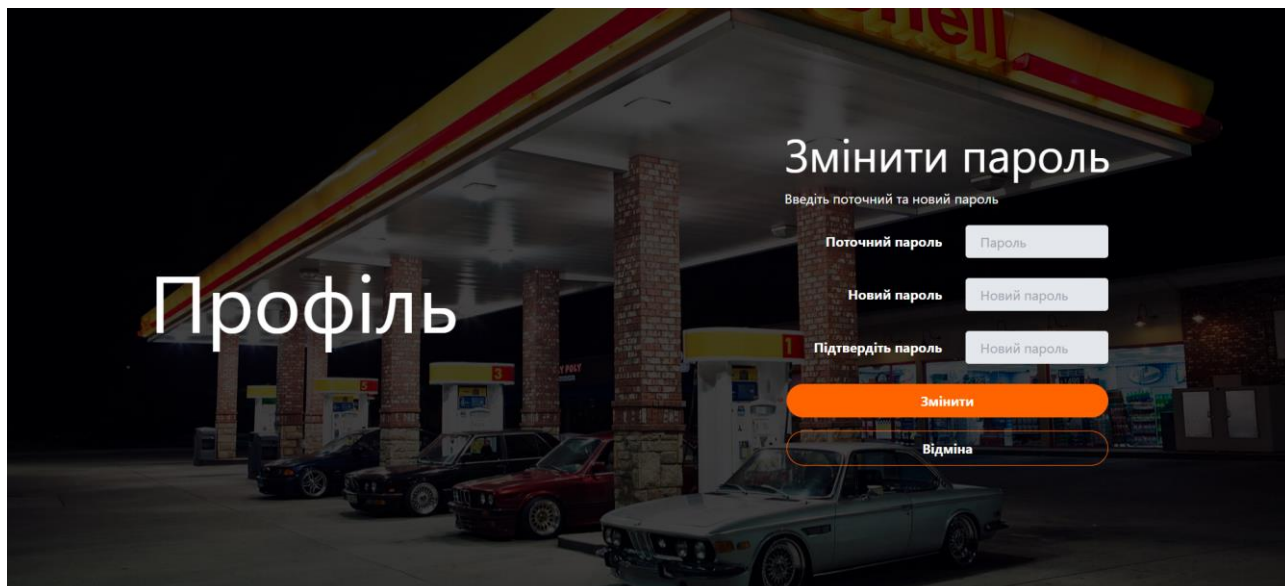


Рисунок 3.19 – Форма зміни паролю постачальника

При натисканні кнопки «Профіль» продавець зможе змінити свій пароль, використовуючи форму зображену на рисунку 3.19.

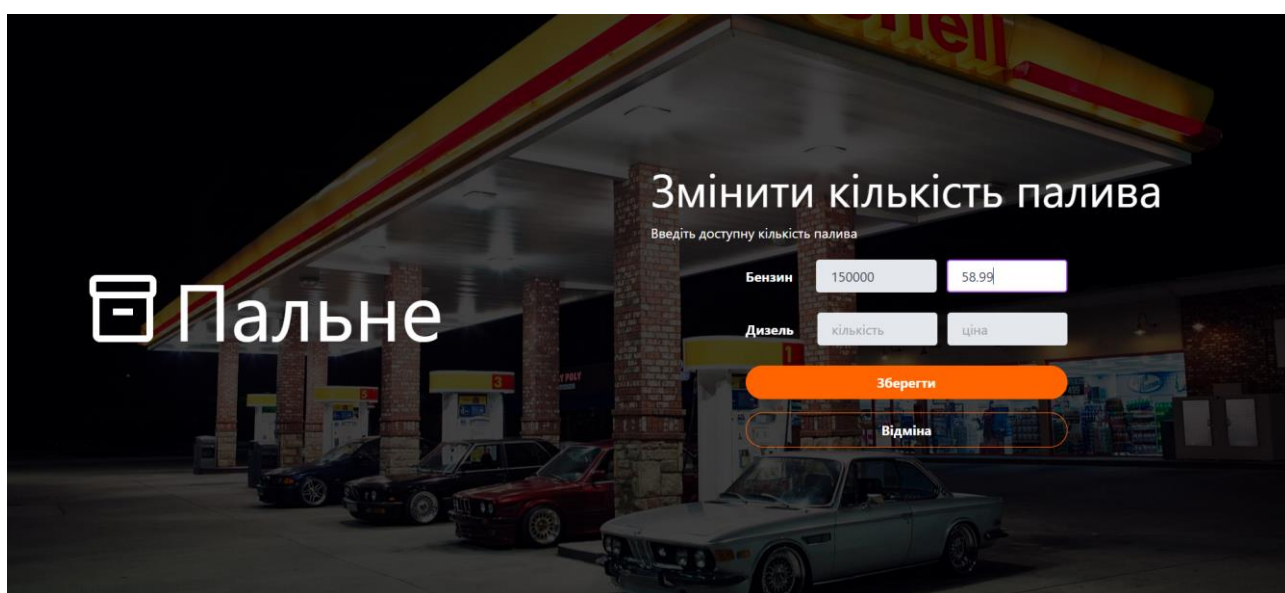


Рисунок 3.20 – Форма редагування кількості та цін палива

При натисканні кнопки «Оновити ціни» продавець може змінити кількість палива на його станції, а також ціни на нього. Даний інтерфейс зображено на рисунку 3.20.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				56
Змн.	Арк.	№ докум.	Підпис	Дата		

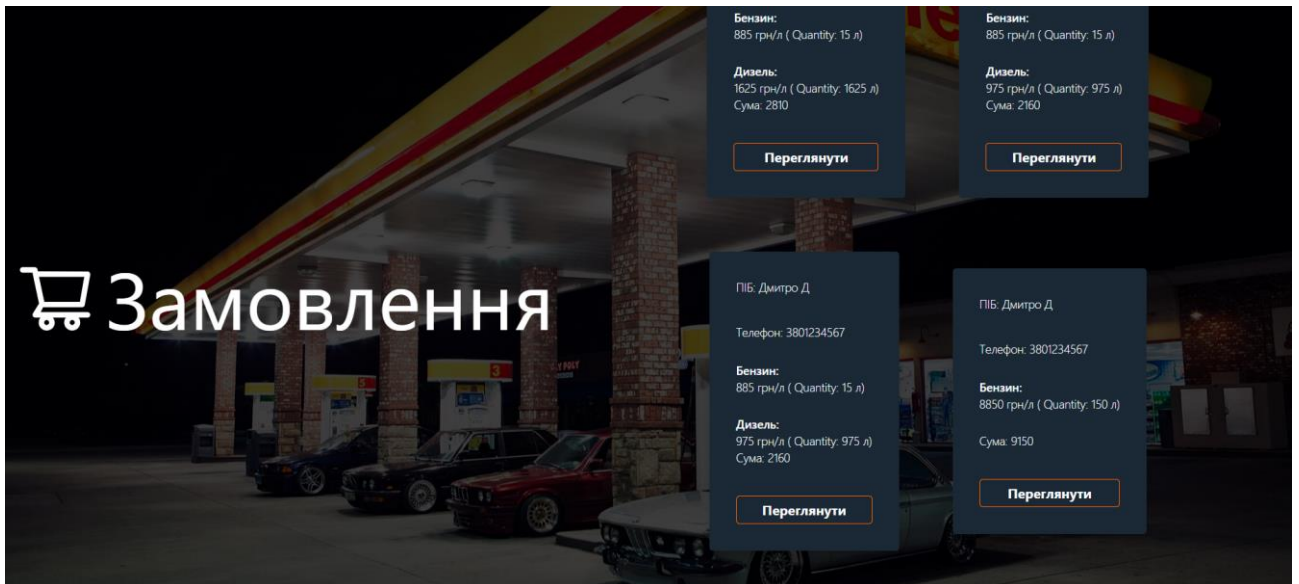


Рисунок 3.21 – Список актуальних замовлень у постачальника

При натисканні кнопки «Замовлення», продавець матиме змогу побачити список його актуальних замовлень та інформацію про клієнта та тип замовлення, що зображено на рисунку 3.21.

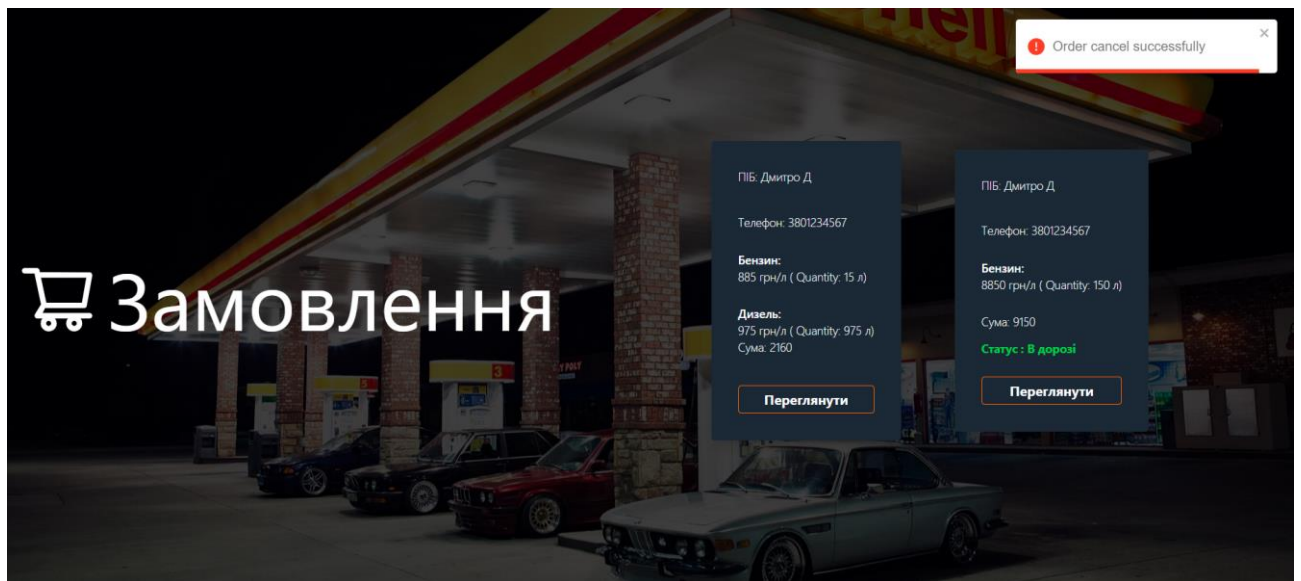


Рисунок 3.22 – Відміна замовлення постачальником

Продавець може переглянути кожне замовлення з місцезнаходженням клієнта та деталями замовлення, а також прийняти його в роботу або ж відмовити клієнту у наданні послуг. Даний функціонал зображено на рисунку 3.22 та 3.23.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				57
Змн.	Арк.	№ докум.	Підпис	Дата		

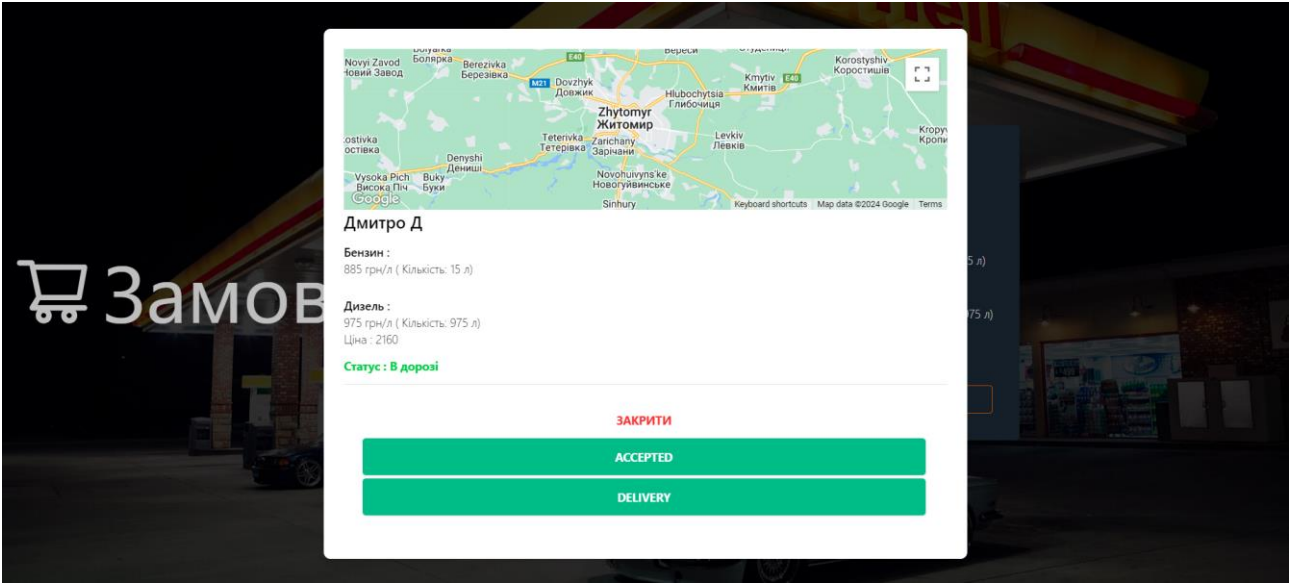


Рисунок 3.23 – Перегляд замовлення онлайн

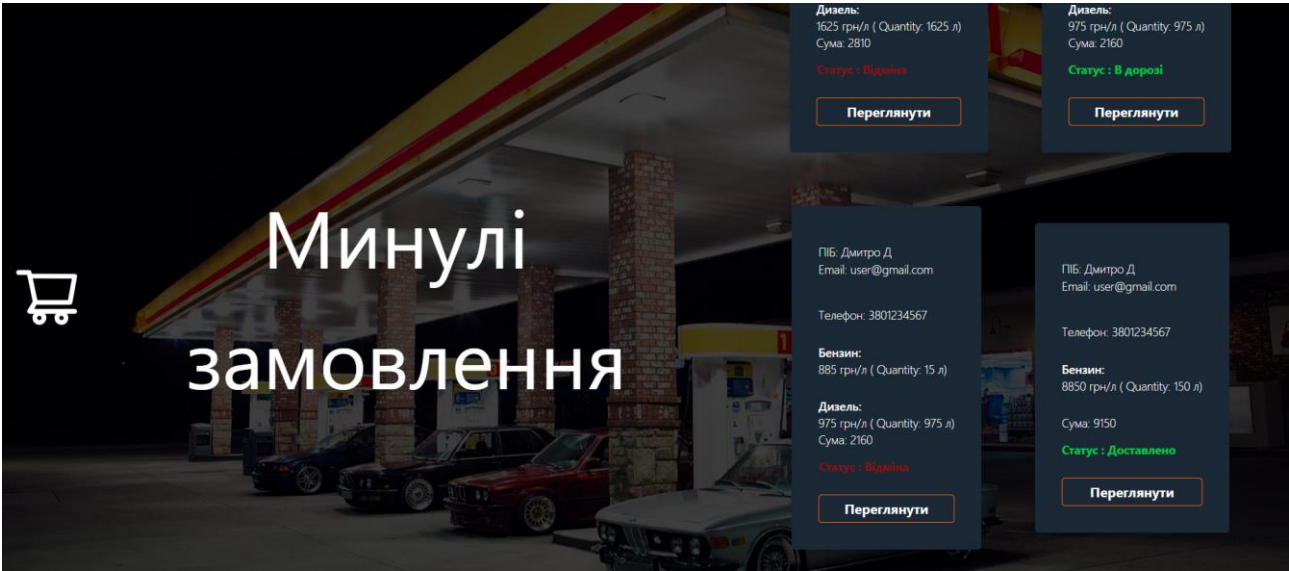


Рисунок 3.24 – Всі замовлення постачальника

При натисканні кнопки «Історія», продавець матиме змогу побачити весь список замовлень, що до нього надходили та їх статуси відповідно. Тут показуються всі замовлення з усіма статусами: вже доставлені, ті, які знаходяться в дорозі, або ж відхилені замовлення. Даний інтерфейс зображено на рисунку 3.24.

3.2 Тестування системи

Для перевірки коректності та надійності роботи веб-додатку було використано техніку мануального тестування. Що стосується виявлення

помилки в коді, було використано апаратне тестування за допомогою інструментів Node.js та React. Критичні помилки були виявлені та виправлені. Веб-додаток має адаптивний дизайн, тому на мобільних телефона працює також коректно та без явних недоліків. Валідація полей, з якими працює користувач, забезпечена використанням шаблонів нашого інструментарію та додаткової перевірки та відловлення невірних даних при роботі з серверною частиною додатку.

Виходячи з вищесказаного, можна говорити про впевненість у коректній роботі веб-додатку під час впровадження його в роботу.

Висновки до третього розділу

У третьому розділі детально розглянуто інтерфейс веб-додатку та його основний функціонал. Показано, як клієнт та продавець можуть використовувати та взаємодіяти з веб-додатком. Для кожної ролі користувача продемонстровано більшість основного функціоналу, з яким він може мати справу, починаючи з реєстрації в системі та закінчуючи переглядом історії замовлень. Так само було продемонстровано унікальні можливості клієнта та продавця, наприклад створення замовлення клієнтом та його відхилення продавцем.

Тестування веб-додатку дало змогу впевнитись у його коректній роботі на будь-яких пристроях завдяки адаптивному дизайну. Валідація полей також дає змогу впевнитись, що користувач не зможе ввести неправильні дані та «крашнути» веб-додаток. Тому можна впевнено сказати, що додаток стабільний та надійний під час його використання.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		59

ВИСНОВОК

В результаті виконання кваліфікаційної роботи було розроблено веб-додаток для онлайн продажу та купівлі палива. Це є результатом аналізу вимог користувачів, існуючих аналогів веб-додатку та детальному аналізу технологій розробки. Під час цих процесів було не лише визначено основні функції та вимоги до системи, але й враховано сучасні напрямки веб-розробки.

Особливу увагу приділено проектуванню архітектури системи та її компонентів, забезпечивши оптимальну взаємодію між ними. Встановлення чітких залежностей та структурування бази даних дозволило забезпечити ефективне управління даними та високу швидкодію системи.

Було проведено аналіз вимог користувачів і надавачів послуг, проведено аналіз вже існуючих рішень, спроектовано архітектуру системи та розроблено базу даних. Також проводилося тестування функціоналу додатку для забезпечення надійності системи та зручності її використання. Практичне значення полягає в тому, що результати роботи можуть бути використані: для оптимізації операційних витрат для компаній, підвищення безпеки операцій, підвищення ефективності бізнесу, підвищення зручності та доступності для клієнтів

У результаті було створено готовий до використання програмний продукт з інтуїтивно зрозумілим інтерфейсом, який відповідає поставленому завданню та потребам користувачів. Було виконано програмне та мануальне тестування розробленого додатку, в результаті можна зробити висновок, що система повністю відповідає заявленим вимогам, не містить помилок та дефектів що впливають на коректність роботи системи.

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		60

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Рейтинг мов програмування 2023 [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://dou.ua/lenta/articles/language-rating-2023/>.
- [2] Рейтинг мов програмування 2024 [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://dou.ua/lenta/articles/language-rating-2024/>.
- [3] JavaScript - краща мова для програмування [Електронний ресурс] – Режим доступу до ресурсу: <https://apeps.kpi.ua/javascript-krashcha-mova-dlia-programuvania>.
- [4] WebStorm Empowering JS Developers [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/webstorm/>.
- [5] 10 Rules of Good UI Design to Follow On Every Web Design Project [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.elegantthemes.com/blog/resources/10-rules-of-good-ui-design-to-follow-on-every-web-design-project>.
- [6] Information Systems for Business and Beyond / D.BOURGEOIS, J. MORTATI, S. WANG, J. SMITH., 2019. – 324 с. – (Biola University).
- [7] MySQL 8.4 Reference Manual [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.4/en/>.
- [8] Модульне тестування [Електронний ресурс] – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/modulne-testuvannya/>.
- [9] Git - Про систему контролю версій [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/book/uk/v2>
- [10] React.JS / Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev>

		Д.ОДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		61

ДОДАТКИ

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import User from "../models/user.js";
import jwt from "jsonwebtoken";
import bcrypt from "bcryptjs";

export const signup=async(req,res)=>{
  const {name,email,password,phone}=req.body;
  try {
    const user=await User.findOne({email});
    if(user){
      return res.status(301).json({message:"User already exists with that email"})
    }

    const hashedPassword=await bcrypt.hash(password,12);

    const newUser=await User.create({
      name,
      email,
      password:hashedPassword,
      phone,

    })
    await newUser.save();
    if(newUser){
      return res.status(201).json({message:"User created successfully"});
    }

  } catch (error) {
    res.status(301).json(error)
  }
}

export const login = async (req, res) => {
  const { email, password } = req.body;
  let user;
  try {
    user = await User.findOne({ email });
    if (!user) {
      return res.status(301).json({ message: "User with that email is not exists" });
    }
    const isValidPassword = await bcrypt.compare(password, user.password);
    if (!isValidPassword) {
      return res.status(300).json({ message: "email and password are incorrect" });
    }
    const authToken = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```
expiresIn: "1hr" });
```

```
    res.status(200).json({ authToken ,userId:user._id});  
  } catch (error) {  
    res.status(400).json(error);  
  }  
}
```

```
export const getUserInfo = async (req,res) =>{  
  try{  
    const user = await User.findOne({_id:req.params.id})  
    if(!user){  
      return res.status(301).json({ message: "User does not exists" });  
    }  
    return res.status(201).json(user)  
  }catch(error){  
    res.status(400).json(error);  
  }  
}
```

```
export const changePassword = async (req,res) =>{  
  const { password,newPassword,userId } = req.body;  
  try {  
    const user = await User.findById(userId);  
  
    if (!user) {  
      return res.status(400).json({ message: "User doesn't Exist" })  
    }  
  
    const isValidPassword = await bcrypt.compare(password, user.password)  
    if (isValidPassword) {  
      const hashedPassword = await bcrypt.hash(newPassword, 12);  
  
      const updatedPassword = await User.updateOne({  
        _id : userId  
      },{  
        $set : {  
          password : hashedPassword  
        }  
      })  
  
      if(!updatedPassword){  
        return res.status(400).json({ message: "Updating Failure" })  
      }  
      return res.status(200).json({ message: "Password Updated Successfully" })  
    }else{
```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

        return res.status(400).json({ message: "Current Password not matching" })
    }

    } catch (error) {
        res.status(400).json(error)
    }
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

import Station from "../models/fuelStation.js";
//import Quantity from "../models/quantity.js";
import jwt from "jsonwebtoken";
import bcrypt from "bcryptjs";

export const registerStation = async (req, res) => {
  const { name, owner, email, password, location, phone } = req.body;
  const { lat, long } = location;
  try {
    const registeredUser = await Station.findOne({
      email
    })
    if (registeredUser) {
      return res.status(400).json({ message: "Already registered" })
    }

    const usedLocation = await Station.findOne({
      "location.lat": lat, "location.long": long
    })

    if (usedLocation) {
      return res.status(400).json({ message: "Location already added" })
    }
    const hashedPassword = await bcrypt.hash(password, 12);
    const newRegisteredUser = await Station.create({
      name,
      owner,
      email,
      password: hashedPassword,
      location,
      phone,
      quantity: {
        petrol: {
          price: 0,
          quantity: 0
        },
        diesel: {
          price: 0,
          quantity: 0
        }
      }
    })
  }

  await newRegisteredUser.save();

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (newRegisteredUser) {
      res.status(201).json({ message: "Registered successfully" })
    }

    } catch (error) {
      res.status(400).json(error)
    }
  }

export const loginStation = async (req, res) => {
  const { email, password } = req.body;
  let loginUser;
  let token;

  try {
    loginUser = await Station.findOne({ email });

    if (!loginUser) {
      return res.status(301).json({ message: "Station with that email does not exists" })
    }

    const isValidPassword = await bcrypt.compare(password, loginUser.password)
    if (!isValidPassword) {
      return res.status(300).json({ message: "email and password are incorrect" });
    }
    token = jwt.sign({ id: loginUser._id }, process.env.JWT_SECRET);
    res.status(200).json({ token ,stationId:loginUser._id})

  } catch (error) {
    res.status(400).json(error)
  }

}

export const updateFuel = async (req,res) =>{
  const { quantity,stationId } = req.body;
  try {
    const user = await Station.findById(stationId);

    if (!user) {
      return res.status(400).json({ message: "User doesn't Exist" })
    }
  }

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

const updatedUser = await Station.updateOne({
  _id : stationId
},{
  $set : {
    quantity : quantity
  }
})

if(!updatedUser){
  return res.status(400).json({ message: "Updating Failure" })
}
return res.status(200).json({ message: "Updated Successfully" })
} catch (error) {
  res.status(400).json(error)
}
}

export const changePassword = async (req,res) =>{
  const { password,newPassword,stationId } = req.body;
  try {
    const user = await Station.findById(stationId);

    if (!user) {
      return res.status(400).json({ message: "User doesn't Exist" })
    }

    const isValidPassword = await bcrypt.compare(password, user.password)
    if (isValidPassword) {
      const hashedPassword = await bcrypt.hash(newPassword, 12);

      const updatedPassword = await Station.updateOne({
        _id : stationId
      },{
        $set : {
          password : hashedPassword
        }
      })

      if(!updatedPassword){
        return res.status(400).json({ message: "Updating Failure" })
      }
      return res.status(200).json({ message: "Password Updated Successfully" })
    }else{
      return res.status(400).json({ message: "Current Password not matching" })
    }
  }
}

```

```

    } catch (error) {
      res.status(400).json(error)
    }
  }
}

```

```

export const getFuelStations = async (req, res) => {
  let stations;
  try {
    stations=await Station.find({})
    if (!stations) {
      return res.status(400).json({ message: "Stations not found" })
    }

    res.status(200).json({stations});
  } catch (error) {
    console.log(error,"err")
    res.status(400).json(error)
  }
}

```

```

export const getStationWithId = async (req, res) => {
  let station;
  try {
    station = await Station.findById(req.params.id);
    if (!station) {
      return res.status(400).json({ message: "Station with that id does not exists" })
    }
    res.status(200).json(station);
  } catch (error) {
    res.status(400).json(error)
  }
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

import Order from "../models/order.js";
import Station from "../models/fuelStation.js"

export const addOrder = async (req, res) => {
  const { address, fuel, method, userId, stationId } = req.body;

  let order;
  try {
    order = await Order.create({
      address,
      fuel,
      method,
      userId,
      stationId,
      isCanceled:{
        status : false,
        message : ""
      },
      isAccepted :{
        status : false,
        message : ""
      },
      isDelivered : {
        status : false,
        message : ""
      }
    });
    if (!order) {
      return res.status(400).json({ message: "Order is not added" });
    }

    const station = await Station.findOne(
      { _id: stationId }
    );
    console.log(stationId)
    let stationQuantity=station.quantity;
    console.log(stationQuantity)
    if(fuel.petrol){
      stationQuantity={
        ...stationQuantity,
        petrol:{
          ...stationQuantity.petrol,
          quantity:stationQuantity.petrol.quantity-fuel.petrol.quantity
        }
      }
    }
  }

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    if(stationQuantity.petrol.quantity<0){
        return res.status(400).json({
            status: "petrol",
            message: "Quantity not available"
        })
    }
}
if (fuel.diesel) {
    stationQuantity = {
        ...stationQuantity,
        diesel: {
            ...stationQuantity.diesel,
            quantity: stationQuantity.diesel.quantity - fuel.diesel.quantity
        }
    }
    if (stationQuantity.diesel.quantity < 0) {
        return res.status(400).json({
            status: "diesel",
            message: "Quantity not available"
        })
    }
}
}

```

```

const updatedStation=await Station.updateOne({
    '_id':station._id,
},{
    $set:{
        quantity: stationQuantity
    }
})
console.log(updatedStation,"qwe")
if (!updatedStation) {

    return res.status(400).json({ message: "Quantity not found" });
}

await order.save();
res
    .status(201)
    .json({ order, userId: order.userId, stationId: order.stationId });
} catch (error) {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

        console.log(error);

        res.status(400).json(error);
    }
};

export const updateOrder=async(req,res)=>{
    const {address,fuel,method}=req.body;
    let order;
    try {
        order=await Order.findByIdAndUpdate(req.params.id,{
            address,
            fuel,
            method
        })

        if(!order){
            return res.status(400).json({message:"Cannot get order with that id"})
        }

        res.status(200).json(order)
    } catch (error) {
        res.status(400).json(error)
    }
}

export const getOrders=async(req,res)=>{
    let orders;
    try {
        orders=await Order.find({});
        if(!orders){
            return res.status(400).json({message:"Orders not found"})
        }

        res.status(200).json(orders);
    } catch (error) {
        res.status(200).json(error);
    }
}

export const getOrderById=async(req,res)=>{

    let order;
    try {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    order=await Order.findById(req.params.id);
    if(!order){
        return res.status(400).json({message:"Order with that id is not exists"});
    }
    res.status(200).json(order);
} catch (error) {
    res.status(400).json(error)
}
}

```

```

export const getOrderById=async(req,res)=>{

```

```

    let order;
    try {
        order=await Order.find({stationId:req.params.id});
        if(!order){
            return res.status(400).json({message:"Order with that id is not exists"});
        }
        res.status(200).json(order);
    } catch (error) {
        res.status(400).json(error)
    }
}

```

```

export const getOrderById=async(req,res)=>{

```

```

    let order;
    try {
        order=await Order.find({userId:req.params.id});
        if(!order){
            return res.status(400).json({message:"Order with that id is not exists"});
        }
        res.status(200).json(order);
    } catch (error) {
        res.status(400).json(error)
    }
}

```

```

export const cancelOrder=async(req,res)=>{
    const {id} =req.body;
    try {

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const updatedOrder=await Order.updateOne({_id:id},
  {$set:{isCanceled:{
    status:true    }}})

if (!updatedOrder){
  return res.status(400).json({message:"Not updated"})
}
res.status(200).json({message:"Order cancel successfully"})
} catch (error) {
  res.status(400).json(error)
}
}

export const acceptOrder = async (req, res) => {
  const { id } = req.body;
  try {
    const updatedOrder = await Order.updateOne({ _id: id },
      {
        $set: {
          isAccepted: {
            status: true,
          }
        }
      })

    if (!updatedOrder) {
      return res.status(400).json({ message: "Not accepted" })
    }
    res.status(200).json({ message: "Order accepted successfully" })
  } catch (error) {
    res.status(400).json(error)
  }
}

export const deliveryOrder = async (req, res) => {
  const { id } = req.body;
  try {
    const updatedOrder = await Order.updateOne({ _id: id },
      {
        $set: {
          isDelivered: {
            status: true,

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

    }
  }
})

if (!updatedOrder) {
  return res.status(400).json({ message: "Not delivered" })
}
res.status(200).json({ message: "Order delivered successfully" })
} catch (error) {
  res.status(400).json(error)
}
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

import Razorpay from "razorpay";
import crypto from 'crypto';

export const initOrder = async (req, res) => {
  const {
    amount
  } = req.body
  try {
    const instance = new Razorpay({
      key_id: process.env.RAZORPAY_KEY_ID,
      key_secret: process.env.RAZORPAY_SECRET,
    });
    const options = {
      amount: amount*100, // amount in smallest currency unit
      currency: "INR",
      receipt: "receipt_order_74394",
    };

    const order = await instance.orders.create(options);

    if (!order) return res.status(500).send("Some error occurred");

    res.json(order);
  } catch (error) {
    res.status(500).send(error);
  }
};

export const verifyOrder = async (req, res) => {
  try {
    // getting the details back from our font-end
    const {
      razorpayPaymentId,
      razorpayOrderId,
      razorpaySignature,
    } = req.body.data;
    const hmac = crypto.createHmac('sha256', process.env.RAZORPAY_SECRET);

    hmac.update(razorpayOrderId + "|" + razorpayPaymentId);
    let generatedSignature = hmac.digest('hex');

    let isSignatureValid = generatedSignature === razorpaySignature;

    // comaparing our digest with the actual signature

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (!isSignatureValid)
        return res.json({ msg: "Transaction not legit!" });

    res.json({
        msg: "Success",
        orderId: razorpayOrderId,
        paymentId: razorpayPaymentId,
    });
} catch (error) {
    console.log(error)
    res.status(500).send(error);
}
};

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

import mongoose from "mongoose";

export const dbconnect=async()=>{
  mongoose.connect(process.env.CONNECTION_URL, { useNewUrlParser: true })
    .then(() => console.log("Mongodb started"))
    .catch((err) => console.log(err))
}

import jwt from "jsonwebtoken";
export const verifyToken = (req, res, next) => {
  const token = req.headers.authorization;
  if (token) {
    jwt.verify(token, process.env.JWT_SECRET, (err) => {
      if (err) return res.sendStatus(403);
      next();
    });
  } else {
    res.sendStatus(401);
  }
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

import mongoose from "mongoose";

const fuelSchema=mongoose.Schema({
  name:{
    type:String,
    required:true,
  },
  owner:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true
  },
  password:{
    type:String,
    required:true
  },
  location:{
    type:{
      lat:{type:Number},
      lng:{type:Number}
    },
    unique:true
  },
  phone:{
    type:Number,
    required:true
  },
  isValid:{
    type:Boolean,
    default:true
  },
  isBlocked:{
    type:Boolean,
    default:false
  },
  quantity:{
    petrol:{
      price:{type:Number},
      quantity: { type: Number }
    },
    diesel: {
      price: { type: Number },

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```
        quantity: { type: Number }  
      }  
    }  
  })
```

```
export default mongoose.model("Station",fuelSchema);
```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import mongoose from "mongoose";

const userSchema=mongoose.Schema({
  name:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true,
    unique:true
  },
  password:{
    type:String,
    required:true
  },
  phone:{
    type:Number,
    required:true
  },
  isValid:{
    type:Boolean,
    default:true
  },
  isBlocked:{
    type:Boolean,
    default:false
  }
})

export default mongoose.model("User",userSchema)

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

import mongoose from 'mongoose';

const orderSchema = mongoose.Schema({
  address: {
    lat: { type: Number, required: true },
    lng: { type: Number, required: true }
  },
  fuel: {
    petrol: {
      price: { type: Number },
      quantity: { type: Number }
    },
    diesel: {
      price: { type: Number },
      quantity: { type: Number }
    }
  },
  method: {
    cash: { type: Number },
    online: {
      transactionID: {
        type: String
      },
      status: {
        type: String
      },
      amount: {
        type: Number
      }
    }
  },
  userId: {
    type: mongoose.Schema.Types.ObjectId,

  },
  stationId: {
    type: mongoose.Schema.Types.ObjectId,

  },
  isCanceled: {
    status: {type:Boolean}
  }
});

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
    isAccepted: {
      status: { type: Boolean }
    },
    isDelivered: {
      status: { type: Boolean },
      message: { type: String }
    },
  },

  })

export default mongoose.model("Order", orderSchema)

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

import express from "express";
import { signup, login, getUserInfo, changePassword } from
"../controllers/userController.js"

const router=express.Router();

router.post('/signup',signup);
router.post('/login',login);
router.put('/changePassword',changePassword)

router.get('/getUserInfo/:id',getUserInfo)
export default router;

import express from "express";
import { changePassword, getFuelStations, getStationWithId, loginStation,
registerStation, updateFuel } from "../controllers/fuelStationController.js";

const router=express.Router();

router.post('/register',registerStation);
router.post('/login',loginStation)
router.put('/changePassword',changePassword);
router.put('/updateFuel',updateFuel);
router.get('/getStations',getFuelStations);
router.get('/:id', getStationWithId)
export default router;

import express from "express";
import { acceptOrder, addOrder,getOrderByUserId, cangleOrder, deliveryOrder,
getOrders,getOrderByFuelStationId} from "../controllers/orderController.js";
const router=express.Router();

router.route('/')
    .post(addOrder)
    .get(getOrders)

router.get('/getOrderByFuelStationId/:id',getOrderByFuelStationId);
router.get('/getOrderByUserId/:id',getOrderByUserId);

router.put('/cancel',cangleOrder);
router.put('/accept',acceptOrder);
router.put('/deliever',deliveryOrder);

export default router;

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

import express from "express";
import { initOrder, verifyOrder } from "../controllers/paymentController.js";

const router=express.Router();

router.post("/order", initOrder);
router.post('/verify',verifyOrder);

export default router;

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

import express from "express";
import cors from "cors";
import dotenv from "dotenv";
import userRoute from "./routes/userRoute.js";
import orderRoute from "./routes/orderRoute.js";
import fuelRoute from "./routes/fuelStationRoute.js";
import paymentRoute from "./routes/paymentRoute.js";

//import quantityRoute from "./routes/quantityRoute.js"
import { dbconnect } from "./database/dbconnection.js";

const app=express();
app.use(cors());
app.use(express.json());
dotenv.config();
dbconnect();

app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Headers', 'Origin,X-Requested-
With,Content-Type,Accept ,Authorization');
    res.setHeader('Access-Control-Allow-Methods', 'GET,POST,PATCH,DELETE');
    next();
})

app.use('/user',userRoute);
app.use('/order',orderRoute);
app.use('/fuel',fuelRoute);
app.use('/payment',paymentRoute);

//app.use('/quantity', quantityRoute)

const PORT=process.env.PORT || 5001;

app.listen(PORT, (req, res) => {
    console.log(`Server started on port:${PORT}`)
})

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

import { Route, Routes, useNavigate } from "react-router-dom";
import Login from "../components/user/login/Login";
import Register from "../components/user/register/Register";
import SimpleMap from "../components/map/Simple";
import Home from "../components/user/home/Home";
import Order from "../components/user/order/Order";
import BookOrder from "../components/user/order/BookOrder";
import Profile from "../components/user/Profile/UpdateProfile";
import FuelHome from "../components/fuelStation/home/Home";
import FuelOrder from "../components/fuelStation/order/Order";
import FuelOrderHistory from "../components/fuelStation/order/OrderHistory";
import UpdateProfile from "../components/fuelStation/Profile/UpdateProfile";
import GetStarted from "../components/getStarted/Index";
import Auth from "../components/user/credentials/Auth";
import FuelAuth from "../components/fuelStation/credentials/Auth";
import FuelLogin from "../components/fuelStation/login/Login";
import FuelRegister from "../components/fuelStation/register/Register";
import UpdateFuel from "../components/fuelStation/inventory/UpdateFuel";
import OrderHistory from "../components/user/order/OrderHistory";
import { toast } from "react-toastify";
import { useEffect } from "react";
function App() {
  return (
    <Routes>
      <Route path="/" element={<GetStarted/>} />
    </Route>
    <Route path="user">
      <Route path="auth" element={<Auth/>} />
      <Route path="register" element={<Register />} />
      <Route path="login" element={<Login />} />
    </Route>
    <Route path="logout" element={<LogoutUser/>} />
    <Route path="" element={<Home />} />
    <Route path="order" element={<Order />} />
    <Route path="bookOrder/:id" element={<BookOrder />} />
    <Route path="Profile" element={<Profile />} />
    <Route path="orderHistory" element={<OrderHistory />} />
  </Route>
  <Route path="seller">
    <Route path="auth" element={<FuelAuth/>} />
    <Route path="register" element={<FuelRegister/>} />
    <Route path="login" element={<FuelLogin/>} />
  </Route>
  <Route path="" element={<FuelHome />} />
  <Route path="order" element={<FuelOrder />} />

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <Route
      path="orderHistory"
      element={<FuelOrderHistory />}
    />
    <Route path="logout" element={<LogoutSeller/>} />
    <Route path="profile" element={<UpdateProfile />} />
    <Route path="update-inventory" element={<UpdateFuel/>} />
  </Route>
  <Route path="map" element={<SimpleMap lat={0.0} lng={0.0} />}></Route>
</Routes>
);
}
const LogoutSeller = () =>{
  const navigate = useNavigate();
  localStorage.removeItem("fuelStation");
  toast.success("Log Out");
  useEffect(()=>{
    navigate("/seller/")
  },[])
}
const LogoutUser = () =>{
  const navigate = useNavigate();
  localStorage.removeItem("user");
  toast.success("Log Out");
  useEffect(()=>{
    navigate("/user/")
  },[])
}
export default App;
export {LogoutSeller};
export {LogoutUser};

```

```

import axios from 'axios';
import authHeader from './auth-header';

const API_URL = 'localhost:5000/user/login';

class UserService {
  getPublicContent() {
    return axios.get(API_URL + 'all');
  }

  getUserBoard() {
    return axios.get(API_URL + 'user', { headers: authHeader() });
  }

  getModeratorBoard() {
    return axios.get(API_URL + 'mod', { headers: authHeader() });
  }

  getAdminBoard() {
    return axios.get(API_URL + 'admin', { headers: authHeader() });
  }
}

export default UserService();

export default function authHeader() {
  const user = JSON.parse(localStorage.getItem("user"));

  if (user && user.accessToken) {
    // return { Authorization: 'Bearer ' + user.accessToken };
    return { "x-auth-token": user.accessToken };
  } else {
    return {};
  }
}

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

import axios from "axios";
import { toast } from "react-toastify";

const API_URL = "http://localhost:5000/";

const signup = (email, password, name, phone) => {
  return axios
    .post(API_URL + "user/signup", {
      email,
      password,
      name,
      phone
    })
    .then((response) => {
      if (response.data.authtoken) {
        localStorage.setItem("user", JSON.stringify(response.data));
      }

      return response;
    });
};

const login = (email, password) => {
  return axios
    .post(API_URL + "user/login", {
      email,
      password,
    })
    .then((response) => {
      console.log(response)
      if (response.data.authtoken) {
        localStorage.setItem("user", JSON.stringify(response.data));
      }
      return response;
    });
};

const logout = () => {
  localStorage.removeItem("user");
};

const getCurrentUser = () => {
  return JSON.parse(localStorage.getItem("user"));
};

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const getUserInfo = (id) => {
  return axios
    .get(API_URL + `user/getUserInfo/${id}`)
    .then((response) => {
      console.log(response)
      return response;
    });
};

const getCurrentFuelStation = () => {
  return JSON.parse(localStorage.getItem("fuelStation"));
};

const fuelStationLogin = (email, password) => {
  return axios
    .post(API_URL + "fuel/login", {
      email,
      password
    })
    .then((response) => {
      console.log(response)
      if (response.data.token) {
        localStorage.setItem("fuelStation", JSON.stringify(response.data));
      }
      return response;
    });
};

const fuelStationRegister = (name,owner,email,password,phone,location) => {
  return axios
    .post(API_URL + "fuel/register", {
      name,
      owner,
      email,
      password,
      phone,
      location
    })
    .then((response) => {
      if (response.data.token) {
        localStorage.setItem("fuelStation", JSON.stringify(response.data));
      }
      return response;
    });
};

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		91

```
const fuelInventoryUpdate = (quantity,stationId) => {
  return axios
    .put(API_URL + "fuel/updateFuel", {
      quantity,stationId
    })
    .then((response) => {
      console.log(response)
      return response;
    });
};
```

```
const getFuelStation = () => {
  return axios
    .get(API_URL + "fuel/getStations")
    .then((response) => {
      return response;
    });
};
```

```
const getFuelStationByID = (id) => {
  return axios
    .get(API_URL + `fuel/${id}`)
    .then((response) => {
      return response;
    });
};
```

```
const postOrder = (userId,stationId,address,fuel,method) =>{
  return axios
    .post(API_URL + `order/`,{
      userId,
      stationId,
      address,
      fuel,
      method
    })
    .then((response) => {
      console.log(response)
      return response;
    });
}
```

```
const cancelOrder = (id) =>{
```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return axios
.put(API_URL + `order/cancel`,{
  id
})
.then((response) => {
  return response;
});
}

const acceptOrder = (id) =>{
  return axios
.put(API_URL + `order/accept`,{
  id
})
.then((response) => {
  return response;
});
}

const deliveryOrder = (id) =>{
  return axios
.put(API_URL + `order/deliever`,{
  id
})
.then((response) => {
  return response;
});
}

const getOrders = (id) =>{
  return axios
.get(API_URL + `order/getOrderByFuelStationId/${id}`)
.then((response) => {
  return response;
});
}

const getUserOrders = (id) =>{
  return axios
.get(API_URL + `order/getOrderByUserId/${id}`)
.then((response) => {
  return response;
});
}

const updateProfilePassword = (userId,password,newPassword) =>{
  return axios

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.put(API_URL + `user/changePassword`,{
  userId,
  password,
  newPassword
})
.then((response) => {
  return response;
});
}

const updateSellerProfilePassword = (stationId,password,newPassword) =>{
  return axios
    .put(API_URL + `fuel/changePassword`,{
      stationId,
      password,
      newPassword
    })
    .then((response) => {
      return response;
    });
}

const loadScript = async(src="https://checkout.razorpay.com/v1/checkout.js")=>{
  return new Promise((resolve) => {
    const script = document.createElement("script");
    script.src = src;
    script.onload = () => {
      resolve(true);
    };
    script.onerror = () => {
      resolve(false);
    };
    document.body.appendChild(script);
  });
}

const displayRazorpay = async(totalDeliveryCharge,setTransactionData)=>{
  const res = await loadScript(
    "https://checkout.razorpay.com/v1/checkout.js"
  );

  if (!res) {
    toast.error("Razorpay SDK failed to load. Are you online?");
    return;
  }
}

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// creating a new order
const result = await axios.post(API_URL + `payment/order`, {
  amount : totalDeliveryCharge
});

if (!result) {
  toast.error("Server error. Are you online?");
  return;
}

// Getting the order details back
const { amount, id: order_id, currency } = result.data;
console.log(amount)
const options = {
  key: "rzp_test_X4JNqOVIvYRBX", // Enter the Key ID generated from the
Dashboard
  currency: currency,
  name: "Fuel Station",
  description: "",
  order_id: order_id,
  handler: async function (response) {
    const data = {
      orderCreationId: order_id,
      razorpayPaymentId: response.razorpay_payment_id,
      razorpayOrderId: response.razorpay_order_id,
      razorpaySignature: response.razorpay_signature,
    };
    console.log(data)

    const result = await axios.post(API_URL + "payment/verify", {data});

    setTransactionData(result.data);
  },

  theme: {
    color: "#61dafb",
  },
};

const paymentObject = new window.Razorpay(options);
paymentObject.open();
}

const authService = {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

signup,
login,
logout,
getCurrentUser,
fuelStationLogin,
getCurrentFuelStation,
fuelStationRegister,
getFuelStation,
getFuelStationByID,
postOrder,
getOrders,
getUserInfo,
cancelOrder,
acceptOrder,
deliveryOrder,
fuelInventoryUpdate,
updateProfilePassword,
updateSellerProfilePassword,
displayRazorpay,
loadScript,
getUserOrders
};

export default authService;

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

import { useNavigate } from "react-router-dom";
import { useEffect, useState } from "react";
import {
  AiOutlineMail,
  AiOutlineEyeInvisible,
  AiOutlineEye,
  AiOutlineLock,
  AiOutlineMobile
} from "react-icons/ai";
import authService from "../../services/auth.service";
import { toast } from "react-toastify";
function Login() {

  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [showPassword, setShowPassword] = useState(false);
  const [error, setError] = useState("");
  const user = authService.getCurrentUser();
  const navigate = useNavigate();

  useEffect(()=>{
    if(user){
      navigate('/user/')
    }
  },[user])

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      await authService.login(email, password).then(
        (response) => {
          toast.success("Login Successful")
          navigate("/user/")
        },
        (error) => {
          toast.error(error.response.data.message);
        }
      );
    } catch (err) {
      console.log(err);
    }
  }

  return (

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<>
<div className="header">
  <h1 className="text-center text-[54px]">Логін</h1>
  <p>Залогуйтесь через email та пароль</p>
</div>
<form class="w-full max-w-sm" onSubmit={handleLogin}>
  <div class="md:flex md:items-center mb-6">
    <div class="md:w-1/3">
      <label
        class="block text-white font-bold md:text-right mb-1 md:mb-0 pr-4"
        for="inline-full-name"
      >
        Email
      </label>
    </div>
    <div class="md:w-2/3">
      <input
        class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"
        id="inline-full-name"
        type="email"
        value={email}
        required
        onChange={(e)=>{
          setEmail(e.target.value)
        }}
        placeholder="abc@gmail.com"
      />
    </div>
  </div>
  <div class="md:flex md:items-center mb-6">
    <div class="md:w-1/3">
      <label
        class="block text-white font-bold md:text-right mb-1 md:mb-0 pr-4"
        for="inline-password"
      >
        Пароль
      </label>
    </div>
    <div class="md:w-2/3 relative flex flex-row">
      <input
        class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 pr-7 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРИНЧАК				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

id="inline-password"
type={`$${(showPassword)?"text":"password"}}`
placeholder="*****"
minLength={8}
required
onChange={(e)=>{
  setPassword(e.target.value)
}}
/>
{showPassword ? (
  <AiOutlineEyeInvisible className="absolute top-3 right-2 text-xl text-
black" onClick={()=>{
    setShowPassword(false)
  }}/>
): (
  <AiOutlineEye className="absolute top-3 right-2 text-xl text-black"
onClick={()=>{
  setShowPassword(true)
  }} />
)
}
</div>
</div>
<div className="actions w-full flex flex-col gap-4">
  <button
    className="bg-[#fe6f2b] hover:bg-[#F59337] text-white font-bold py-2 px-
4 rounded-full"
    >
    Логін
  </button>
  <button
    className="bg-transparent border border-[#fe6f2b] hover:bg-[#F59337]
text-white font-bold py-2 px-4 rounded-full"
    onClick={((e)=>{
      e.preventDefault();
      navigate('../register')
    })}
    >
    Реєстрація
  </button>
  <button
    className="bg-transparent mb-5 border border-[#fe6f2b] hover:bg-
[#F59337] text-white font-bold py-2 px-4 rounded-full"
    onClick={((e)=>{
      e.preventDefault();

```

```

        navigate('/seller/')
      })}
    >
      Продавець
    </button>
  </div>
</form>
</>

);
}
export default Login;

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { useNavigate } from "react-router-dom"
import GetStarted from "../assets/images/getStarted.png"
function Index(){
  const navigate = useNavigate();
  return(
    <div className="w-screen h-screen bg-[#252422] flex flex-col lg:flex-row
justify-evenly items-center">
      <div className="h-[45%]">
        <img src={GetStarted} alt={"Get Started"}/>
      </div>
      <div className="text-white flex flex-col gap-10">
        <div className="header">
          <h1 className="text-center text-[54px]">
            Вітаємо
          </h1>
          <p>
            Купуй пальне, не виходячи з дому.
          </p>
        </div>
        <div className="actions w-full flex flex-col gap-4">
          <button className="bg-[#fe6f2b] hover:bg-[#F59337] text-white font-
bold py-2 px-4 rounded-full"
onClick={()=>{
  navigate('user/auth/login')
}}
>
            Клієнт
          </button>
          <button className="bg-transparent border border-[#fe6f2b] hover:bg-
[#F59337] text-white font-bold py-2 px-4 rounded-full"
onClick={()=>{
  navigate('seller/auth/login')
}}
>
            Продавець
          </button>
        </div>
      </div>
    </div>
  )
}
export default Index

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { useNavigate } from "react-router-dom";
import LoginLight from "../../assets/images/loginLight.jpg";
import { MdOutlineInventory2 } from "react-icons/md"
import { useEffect, useState } from "react";
import { toast } from "react-toastify";
import authService from "../../services/auth.service";
function UpdateFuel() {
    const [petrolQuantity,setPetrolQuantity] = useState(null);
    const [petrolPrice,setPetrolPrice] = useState(null);

    const [dieselQuantity,setDieselQuantity] = useState(null);
    const [dieselPrice,setDieselPrice] = useState(null);

    const navigate = useNavigate()
    const fuelStation = authService.getCurrentFuelStation();

    useEffect(()=>{
        if(!fuelStation){
            navigate('../auth/login')
        }
    },[])
    const updateInventory = async () =>{
        const quantity = {

        }

        if(petrolQuantity){
            const petrol = {
                "price" : petrolPrice,
                "quantity" : petrolQuantity
            }
            quantity.petrol = petrol;
        }
        if(dieselPrice){
            const diesel = {
                "price" : dieselPrice,
                "quantity" : dieselPrice
            }
            quantity.diesel = diesel;
        }
        try {
            await authService.fuelInventoryUpdate(quantity,fuelStation.stationId).then(
                (response) => {
                    toast.success(response.data.message)
                    navigate('../')
                }
            ),

```

		Д.О.ДЗЮБЕНКО			ICT.KP.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        (error) => {
            toast.error(error.response.data.message);
        }
    );
} catch (err) {
    console.log(err);
}
}

const validation = () =>{
    if(petrolPrice !== "" && petrolQuantity ==="" ){
        toast.warning("Please Fill in Petrol Quantity")
        return false;
    }

    if(petrolPrice === "" && petrolQuantity !=="" ){
        toast.warning("Please Fill in Petrol Price")
        return false;
    }

    if(dieselPrice !== "" && dieselQuantity ==="" ){
        toast.warning("Please Fill in Diesel Quantity")
        return false;
    }

    if(dieselPrice === "" && dieselQuantity !=="" ){
        toast.warning("Please Fill in Diesel Price")
        return false;
    }
    return true
}

const onHandleSubmit = (e) =>{
    e.preventDefault();
    const boolean = validation();
    if(boolean){
        updateInventory()
    }
}

useEffect(()=>{

    },[petrolPrice,petrolQuantity,dieselPrice,dieselQuantity])
return (
    <div
        className="w-screen h-screen flex flex-col justify-around items-center
lg:md:flex-row"

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

style={{
  backgroundImage: `linear-gradient(45deg,rgba(0,0,0, 0.75),rgba(0,0,0,
0.75)),url(${LoginLight})`,
  backgroundPosition: `50% 50%`,
  backgroundSize: `cover`,
  backgroundRepeat: "no-repeat",
}}
>
<div className="text-white p-3 text-center text-[54px] flex flex-row justify-
center items-center gap-3 whitespace-break-spaces font-sans lg:text-[96px] md:text-
[74px]">
  <MdOutlineInventory2/>
  <h1>Пальне</h1>
</div>
<div className="w-[100%] text-white h-[100%] justify-center gap-5 lg:w-1/2
items-center flex flex-col flex-wrap overflow-scroll">
  <div className="header">
    <h1 className="text-center text-[54px]">Змінити кількість палива</h1>
    <p>Введіть доступну кількість палива</p>
  </div>
  <form class="w-full max-w-sm" onSubmit={onHandleSubmit}>
    <div class="gap-3 md:flex md:items-center mb-6">
      <div class="mb-3 lg:mb-0">
        <label
          class="block text-white font-bold md:text-right mb-1 md:mb-0 pr-4"
          for="inline-petrol"
        >
          Бензин
        </label>
      </div>
      <div class="mb-3 lg:mb-0">
        <input
          class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"
          id="inline-petrol"
          type="number"
          value={petrolQuantity}

          onChange={(e)=>{
            setPetrolQuantity(e.target.value)
          }}
          placeholder="кількість"
        />
      </div>
    </div>
  </form>
</div>

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div class="">
  <input
    class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"
    id="inline-petrol"
    type="number"
    value={petrolPrice}

    onChange={(e)=>{
      setPetrolPrice(e.target.value)
    }}
    placeholder="ціна"
  />
</div>
</div>
<div class="gap-3 md:flex md:items-center mb-6 ">
  <div class="mb-3 lg:mb-0">
    <label
      class="block text-white font-bold md:text-right mb-1 md:mb-0 pr-4"
      for="inline-diesel"
    >
      Дизель
    </label>
  </div>
  <div class="mb-3 lg:mb-0">
    <input
      class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"
      id="inline-diesel"
      type="number"
      value={dieselQuantity}

      onChange={(e)=>{
        setDieselQuantity(e.target.value)
      }}
      placeholder="кількість"
    />
  </div>
</div>
<div class="">
  <input
    class="bg-gray-200 appearance-none border-2 border-gray-200 rounded w-
full py-2 px-4 text-gray-700 leading-tight focus:outline-none focus:bg-white
focus:border-purple-500"

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        id="inline-diesel"
        type="number"
        value={dieselPrice}

        onChange={(e)=>{
            setDieselPrice(e.target.value)
        }}
        placeholder="ціна"
    />
</div>
</div>
<div className="actions flex flex-col gap-4">
    <button
        className="bg-[#fe6f2b] hover:bg-[#F59337] text-white font-bold py-2 px-
4 rounded-full"
    >
        Зберегти
    </button>
    <button
        className="bg-transparent border border-[#fe6f2b] hover:bg-[#F59337]
text-white font-bold py-2 px-4 rounded-full"
        onClick={((e)=>{
            e.preventDefault();
            navigate('./')
        })}
    >
        Відміна
    </button>
</div>
</form>
</div>
</div>
);
}
export default UpdateFuel;

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { useEffect, useState } from "react";
import SimpleMap from "../map/Simple";
import { getDistance } from "geolib";
import { Navigate, useNavigate } from "react-router-dom";
import authService from "../services/auth.service";
import OrderPreview from "../modal/OrderPreview";
import { toast } from "react-toastify";

function ListOrder({ order,setLoading }) {
  const { address, fuel, isAccepted,isCanceled,isDelivered,method,userId,_id} =
order;
  const [showModal, setShowModal] = useState(false);
  const navigate = useNavigate();
  useEffect(()=>{
    getUserInfo()
  },[])

  const [userInfo,setUserInfo] = useState(null);
  const getUserInfo = async () =>{
    try {
      await authService.getUserInfo(userId).then(
        (response) => {
          setUserInfo(response.data);
        },
        (error) => {
          console.log(error.response);
        }
      );
    } catch (err) {
      console.log(err);
    }
  }

  const cancelOrder = async () =>{
    try {
      await authService.cancelOrder(_id).then(
        (response) => {
          toast.error(response.data.message)
          setLoading(true);
        },
        (error) => {
          console.log(error.response);
        }
      );
    } catch (err) {

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    console.log(err);
  }
}

```

```

const deliveryOrder = async () =>{
  try {
    await authService.deliveryOrder(_id).then(
      (response) => {
        toast.success(response.data.message);
        setLoading(true);
      },
      (error) => {
        console.log(error.response);
      }
    );
  } catch (err) {
    console.log(err);
  }
}

```

```

const acceptOrder = async () =>{
  try {
    await authService.acceptOrder(_id).then(
      (response) => {
        toast.info(response.data.message)
        setLoading(true);
      },
      (error) => {
        console.log(error.response);
      }
    );
  } catch (err) {
    console.log(err);
  }
}

```

```

const renderedUserInfo = (userInfo)?
<>
<p className="text-grey-dark font-thin text-sm leading-normal text-white">
  ПІБ: {userInfo.name}
</p>
<p className="text-grey-dark font-thin text-sm leading-normal text-white">
  <br />
  Телефон: {userInfo.phone}
</p>

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				108
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<p className="text-grey-dark font-thin text-sm leading-normal text-white">
</p>
</>
:null

```

```

const renderedOrderInfo = <>
<div className="place-self-start">
  {(fuel.petrol)?

```

```

    <div className="text-sm text-white font-semibold">
      <p>Бензин: </p>
      <p className="text-sm font-thin">
        {fuel.petrol.price} грн/л ( Quantity: {fuel.petrol.quantity} л)
      </p>
    </div>:null}
    <br/>
    {(fuel.diesel)?
    <div className="text-sm text-white font-semibold">
      <p>Дизель: </p>
      <p className="text-sm font-thin">
        {fuel.diesel.price} грн/л ( Quantity: {fuel.diesel.quantity} л)
      </p>
    </div>:null}
    <div className="text-sm text-white font-semibold">
      <p className="text-sm text-white font-thin">
        Сума: {(method.cash)?method.cash:method.online.amount}
      </p>
    </div>
  </div>
  <div className="text-sm font-semibold">
    <p className={` ${isAccepted.status && !isDelivered.status}? " text-
[#32CD32] font-bold ": "hidden" }`>
      Статус : В дорозі
    </p>
    <p className={` ${isCanceled.status}? " text-red-900 font-bold ":
"hidden" }`>
      Статус : Відміна
    </p>
    <p className={` ${isDelivered.status}? " text-[#32CD32] font-bold ":
"hidden" }`>
      Статус : Доставлено
    </p>
  </div>

```

```

<p className="text-grey-dark font-thin text-sm leading-normal text-white">

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</p>
</>
console.log(isAccepted)
return (
  <div className={`shadow-lg gap-3 rounded m-8 p-8 flex
${(isAccepted.status)?" text-white":""} bg-gray-800`} >
    <div className="w-full lg: md: flex flex-col gap-3 ">
      {/* <h3 className="text-orange text-xl font-semibold ">{""}</h3> */}
      {renderedUserInfo}
      {renderedOrderInfo}
      <button className={`bg-transparent border-[#fe6f2b] hover:border-transparent
hover:bg-[#fe6f2b] font-bold text-white py-1 border rounded`} onClick={()=>{
        setShowModal(true)
      }}>
        Переглянути
      </button>
      {
        showModal?
        <OrderPreview order={order} userInfo={userInfo}
setOnClose={setShowModal} setOnDelivery={()=>{
          deliveryOrder()
          setShowModal(false)
        }}
setOnCancel={()=>{
          cancelOrder()
          setShowModal(false)
        }}
setOnApply={
          ()=>{
            acceptOrder()
            setShowModal(false)
          }
        }/>
        :null
      }
    </div>
  </div>
);
}
export default ListOrder;

```

		Д.О.ДЗЮБЕНКО			ІСТ.КР.Б – 126 – 02 – ПЗ	Арк.
		А.І. ОРІНЧАК				110
Змн.	Арк.	№ докум.	Підпис	Дата		