

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Додаток для відстеження стану технічного
обслуговування автомобілів»**

Виконала студентка групи ІПЗ-21-2
МИРОНОВА Вікторія Леонідівна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МИРОНОВА Вікторія Леонідівна**

Керівник роботи: **ЧИЖМОТРИЯ Олексій Володимирович**

Тема роботи: **«Додаток для відстеження стану технічного обслуговування автомобілів»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: процес автоматизації управління замовленнями та технічним обслуговуванням автомобілів на СТО, методи та засоби розробки інформаційної системи для планування, керування ресурсами і взаємодії користувачів в рамках процесів СТО.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	27.02.2025р.	27.02.2025р.
2	Чижмотря О.В.	27.03.2025р.	27.03.2025р.
3	Чижмотря О.В.	27.04.2025р.	27.04.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота складається з програмного комплексу веб-додатку для відстеження технічного обслуговування (ТО) автомобілів та пояснювальної записки. Пояснювальна записка містить текстову частину, викладену на 67 сторінках друкованого тексту, має 41 ілюстрацію, 16 таблиць, 5 додатків та список використаних джерел із 30 найменувань. Загальний обсяг роботи становить 81 сторінка.

Метою роботи є розробка програмного забезпечення для автоматизації процесів обліку та контролю технічного обслуговування транспортних засобів на станціях технічного обслуговування (СТО). Система призначена для підвищення ефективності ведення замовлень, зберігання інформації про клієнтів і співробітників, формування документів, обліку запчастин і обчислення вартості послуг.

У роботі розглянуто підходи до розробки програмного забезпечення для автоматизації діяльності станції технічного обслуговування автомобілів. Проаналізовано аналогічні системи, обґрунтовано вибір архітектури та технологій розробки, спроектовано структуру застосунку та бази даних. Надано опис функціональних модулів системи, принципів їх взаємодії та основних сценаріїв застосування. Розроблено інтерфейс користувача, реалізовано базовий функціонал додатку та проведено його тестування.

Ключові слова: БАЗА ДАНИХ, ЗАМОВЛЕННЯ, ОБСЛУГОВУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СТО.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Миронова В.Л.			Додаток для відстеження стану технічного обслуговування автомобілів	Літ.	Арк.	Аркушів
Керівник		Чижмотря О. В.					3	81
						ФІКТ Гр. ІПЗ-21-2		
Рецензент		Лобанчикова Н.М.						
Зав. каф.		Вакалюк Т.А						

ABSTRACT

The bachelor's qualification work consists of a software system in the form of a web application for tracking vehicle maintenance and an explanatory report. The explanatory report contains the textual part, presented on 67 pages of printed text, includes 41 illustrations, 16 tables, 5 appendices, and a list of references with 30 sources. The total volume of the work amounts to 81 pages.

The purpose of this work is to develop software for automating the processes of accounting and controlling the maintenance of vehicles at service stations (STOs). The system is designed to improve the efficiency of order management, store information about clients and employees, generate documents, manage spare parts inventory, and calculate the cost of services.

The work explores approaches to developing software for automating the operations of car service stations. Similar systems are analyzed, the choice of architecture and development technologies is justified, and the structure of the application and the database is designed. A description of the system's functional modules, the principles of their interaction, and the main usage scenarios is provided. A user interface has been developed, the basic functionality of the application has been implemented, and testing has been conducted.

Keywords: DATABASE, ORDER, MAINTENANCE, SOFTWARE, SERVICE STATION.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижевотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ ДЛЯ ВІДСТЕЖЕННЯ ТО АВТОМОБІЛІВ	10
1.1. Постановка задачі.....	10
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектур вебдодатку.....	16
Висновки до розділу 1	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ	22
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	22
2.2. Розробка бази даних системи	28
2.3. Проектування та реалізація алгоритмів роботи системи	34
2.4. Реалізація додатку для відстеження технічного обслуговування автомобілів	38
Висновки до розділу 2	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ	43
3.1. Порядок встановлення та налаштування параметрів системи	43
3.2. Структура інтерфейсу веб-орієнтованої системи для відстеження технічного обслуговування автомобілів.....	45
3.3. Тестування роботи програмного комплексу	60
Висновки до розділу 3	62
ВИСНОВКИ.....	63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	65
ДОДАТКИ.....	68

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних;

ПЗ – Програмне забезпечення;

СТО – Станції технічного обслуговування;

ТО – Технічне обслуговування.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Сучасний рівень розвитку інформаційних технологій створює передумови для автоматизації практично всіх сфер діяльності людини, зокрема у галузі обслуговування та ремонту автомобільного транспорту. Підвищення якості надання послуг на станціях технічного обслуговування (СТО), зменшення часу обслуговування клієнтів, ефективне управління персоналом, матеріальними ресурсами та складом — все це потребує використання сучасних комп'ютеризованих систем. Використання спеціалізованого програмного забезпечення допомагає автоматизувати ці процеси та збільшити ефективність роботи підприємств. Проте, багато СТО й досі використовують паперову документацію або застаріле програмне забезпечення, що значно ускладнює процес обліку, планування та взаємодії між співробітниками і клієнтами.

Актуальність теми дослідження обумовлена потребою у розробці ефективного інструменту, який дозволяє автоматизувати облік замовлень, призначення відповідальних осіб, ведення клієнтської бази, калькуляцію вартості послуг, друк супровідної документації, а також управління складом запчастин. В умовах постійного зростання вимог до якості обслуговування та ефективного управління ресурсами, впровадження програмного забезпечення такого типу є надзвичайно важливим та доцільним.

Метою даної випускної роботи є розробка алгоритмів роботи та реалізація програмного комплексу для автоматизації процесів на станції технічного обслуговування автомобілів засобами HTML, CSS, JavaScript а також розробка системи адміністрування за допомогою мови програмування C# із використанням .NET Framework та бази даних MS SQL.

Завдання роботи полягають у наступному:

- проведення аналізу задач управління процесами технічного обслуговування автомобілів;
- дослідити існуючі аналоги програмних продуктів;
- обґрунтування вибору архітектури та інструментальних засобів;

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

- обрати програмні засоби реалізації;
- розробка інформаційної моделі системи та структури бази даних;
- проектування логіки роботи функціональних модулів системи;
- спроектувати базу даних для збереження інформації про сервіс та користувачів;
- побудова алгоритмів взаємодії користувача із системою;
- реалізація користувацького інтерфейсу програми;
- забезпечення можливості адміністрування системи;
- тестування програмного продукту та аналіз отриманих результатів.

Об’єктом дослідження є процес автоматизації управління замовленнями та технічним обслуговуванням автомобілів на СТО.

Предметом дослідження є методи та засоби розробки інформаційної системи для планування, керування ресурсами і взаємодії користувачів в рамках процесів СТО.

Методи дослідження. У роботі використано методи об’єктно-орієнтованого проектування, структурування даних, а також технології програмування з використанням мови C#, фреймворку .NET, моделювання реляційної бази даних MS SQL. Для побудови інтерфейсу та реалізації логіки додатку використано підхід багаторівневої архітектури.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто загальні принципи побудови систем автоматизації, аналіз аналогів та вибір технологій. У другому розділі описано проектування системи, бази даних та логіки взаємодії. У третьому розділі наведено інтерфейс користувача, тестування та результати роботи додатку.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ ДЛЯ ВІДСТЕЖНЕННЯ ТО АВТОМОБІЛІВ

1.1. Постановка задачі

На сьогоднішній день стан транспортних засобів має надзвичайно важливе значення для гарантування безпеки та надійності при час їх експлуатації. Особливо це стосується для станцій технічного обслуговування (СТО), які займаються діагностикою, обслуговуванням і ремонтом транспортних засобів. Своєчасне виявлення несправностей і планування профілактичного обслуговування дозволяє покращити якість послуг та запобіганню серйозних поломок.

Технічне обслуговування [1] — це не просто обов’язкова процедура, а вирішальний аспект експлуатації авто, який безпосередньо впливає на безпеку водіїв та довговічність транспортного засобу. Однак сучасні СТО часто стикаються з низкою труднощів: необхідно вести облік замовлень, розподіляти навантаження між працівниками, контролювати залишки запчастин і підтримувати зв’язок з клієнтами.

Використання сучасних технологій дозволяє автоматизувати частину цих процесів. Наприклад, спеціалізований веб додаток може значно полегшити моніторинг технічного стану автомобілів, покращити зв’язок між працівниками СТО та клієнтами, а також мінімізувати виникнення помилок під час обслуговування.

СТО часто зіштовхуються з низкою проблем, які суттєво впливають на ефективність їхньої роботи та якість обслуговування клієнтів. Однією з основних труднощів є відсутність єдиної цифрової системи обліку. Інформацію про клієнтів, автомобілі, виконані роботи та замовлення часто записують вручну, що не лише займає багато часу, але й призводить до помилок, втрати важливих даних або плутанини в замовленнях.

Також важко здійснювати ефективне планування роботи сервісу. Без автоматизації, працівникам складно передбачити, коли і які автомобілі потребуватимуть

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

обслуговування, які ресурси будуть потрібні, чи достатньо запасних частин на складі. Це може призводити до простоїв, затримок і незадоволення клієнтів.

Крім того, працівники СТО не завжди мають зручний доступ до повної історії обслуговування кожного автомобіля. Відсутність такої інформації ускладнює прийняття оперативних рішень, у тому числі щодо повторних ремонтів, гарантійного обслуговування чи заміни запчастин. Усе це сповільнює робочі процеси та створює додаткове навантаження на персонал.

Для подолання зазначених проблем доцільним є створення спеціалізованого веб-додатку для СТО, який допоможе налагодити роботу сервісу. Такий інструмент дозволить у зручній формі відстежувати технічний стан автомобілів, забезпечить простий доступ до інформації як для майстрів, так і для клієнтів. Система зможе автоматично генерувати необхідні звіти, а також пропонувати рекомендації щодо обслуговування чи ремонту на основі зібраних даних.

Метою даного проєкту є розробка веб-додатку, який допоможе автоматизувати процеси на СТО, покращити якість обслуговування, підвищити ефективність роботи персоналу та забезпечити зручність для клієнтів.

Основними вхідними даними в системі будуть: VIN-коди, марка та модель авто, пробіг, історія технічного обслуговування, а також інформація про клієнтів. Система повинна бути простою у використанні, мати зручний інтерфейс для введення та перегляду інформації, а також дозволяти швидко знаходити потрібні дані.

Додаток розроблений таким чином, щоб охопити всі основні потреби станції технічного обслуговування в повсякденній роботі. Він дає змогу співробітникам СТО проходити реєстрацію та авторизацію. Завдяки системі можна зручно планувати замовлення, стежити за завантаженням ресурсів, розподіляти роботу між ними та контролювати хід виконання.

Кожне нове замовлення можна швидко створити в кілька кліків. До нього можна додати клієнта — як обрати з уже наявної бази, так і створити нового. Замовлення автоматично закріплюється за відповідальним менеджером, який контролює його виконання.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Система дозволяє відслідковувати статус кожного замовлення в режимі реального часу. Також передбачена можливість додавати нових співробітників, що важливо для масштабованості сервісу. Після завершення обслуговування додаток розраховує загальну вартість послуг, автоматично формує документи, які можна зберегти або роздрукувати — наприклад, акт виконаних робіт.

Окремо реалізовано облік запчастин. Система фіксує наявність деталей на складі та допомагає уникнути нестач. Також ведеться перелік усіх доступних робіт і послуг із зазначенням вартості, що забезпечує прозорість для клієнтів і зручність для персоналу.

1.2. Аналіз аналогів програмного продукту

На сучасному ринку існує чимало рішень, спрямованих на автоматизацію діяльності станцій технічного обслуговування (СТО). Ці системи відрізняються функціональністю, зручністю використання, можливістю інтеграції з іншими сервісами, а також вартістю. Аналіз існуючих аналогів дає змогу визначити як базові потреби ринку, так і слабкі місця наявних продуктів, що дозволяє створити конкурентоздатне рішення.

Серед найпопулярніших аналогів програмних продуктів у цій сфері можна виокремити наступні: AutoRepair Cloud, RemOnline, SmartCRM та Shop-Ware. Саме ці продукти вважаються найбільш функціональними, поширеними та придатними для аналізу.

Під час аналізу подібних програмних рішень важливо звертати увагу на кілька ключових аспектів, які впливають на зручність та ефективність їх використання. Зокрема, слід враховувати, наскільки зручним і зрозумілим є інтерфейс, чи відповідає дизайн сучасним вимогам і очікуванням користувача. Важливо також оцінити функціональність системи, чи охоплює вона всі необхідні процеси для СТО.

Не менш значущим є питання адаптивності. Наскільки добре система працює на різних пристроях. Зручність налаштування також має велике значення, адже користувачі повинні мати змогу швидко пристосувати систему до своїх потреб без складних технічних дій.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Додатковими перевагами є наявність інструментів для складського обліку, можливість формування та друку шаблонних документів. І, звичайно, важливою умовою є наявність локалізації для українського ринку – як у плані мови, так і з урахуванням специфіки документообігу.

Першим прикладом є AutoRepair Cloud [2] – зручне хмарне рішення для організації роботи СТО (рис.1.1.). Програма має простий у використанні веб-інтерфейс англійською мовою, через який можна працювати з базою клієнтів, замовленнями, складом і фінансами. Важливою перевагою є інтеграція з календарями, поштою та іншими корисними сервісами. Водночас відсутність української локалізації може створювати труднощі для користувачів в Україні

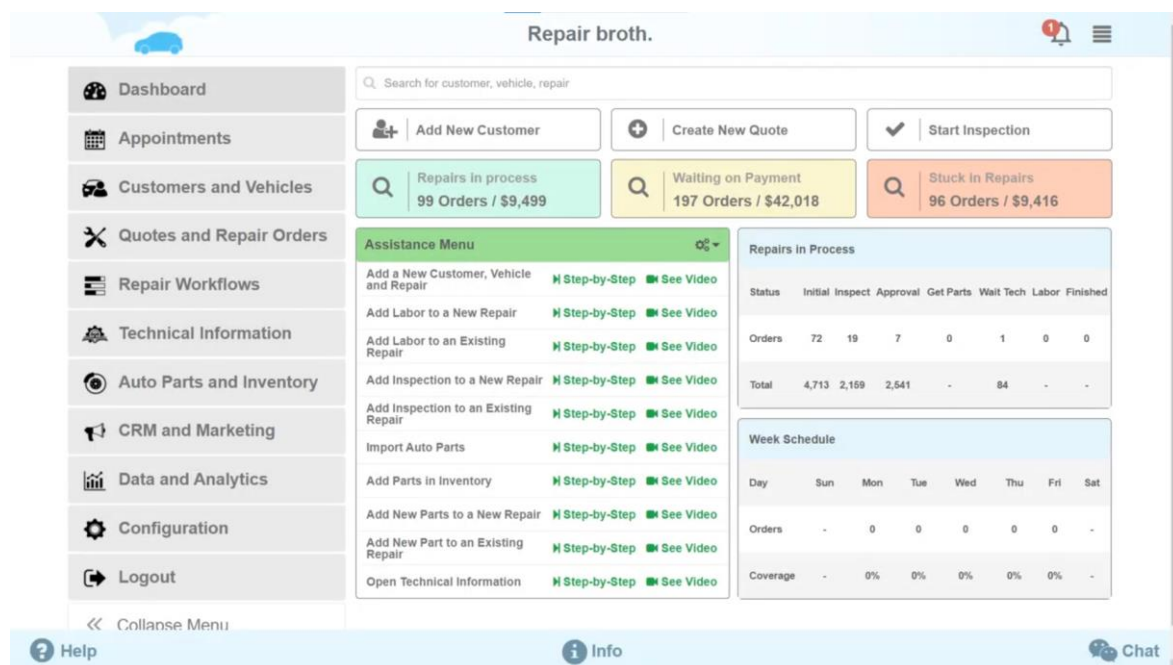
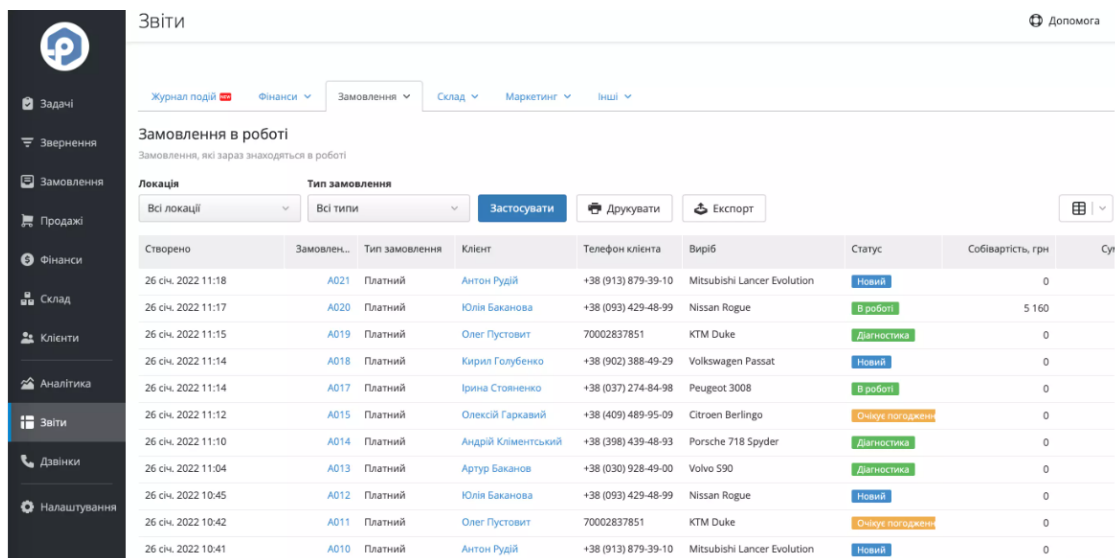


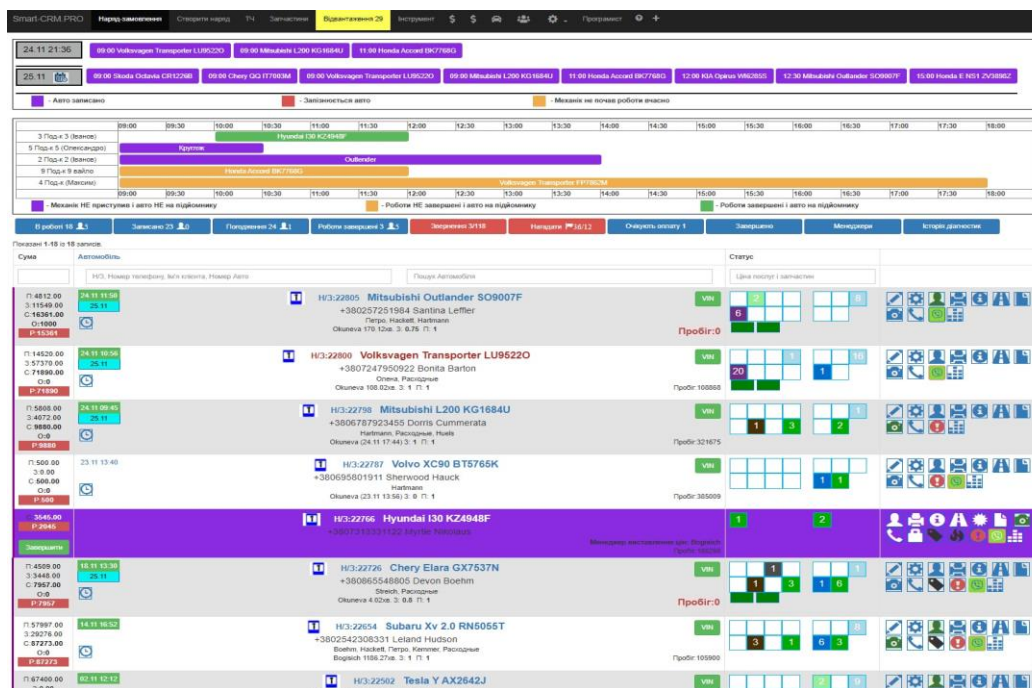
Рисунок 1.1. AutoRepair Cloud

Другим прикладом є RemOnline [3] – програма для автосервісу та СТО (рис.1.2.). Це хмарна програма, яка працює через браузер і має сучасний інтерфейс з підтримкою української мови. Серед основних можливостей: ведення замовлень, управління складом, облік фінансів, CRM для роботи з клієнтами, формування звітів та інтеграції з іншими сервісами. Попри широкий функціонал, RemOnline більше підходить для середнього бізнесу через платну модель підписки. Для новачків інтерфейс може здатися трохи перевантаженим, тож знадобиться трохи часу, щоб розібратися, як усе працює.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13



SmartCRM [4] — спеціалізована CRM-система для автосервісів (рис.1.3.). Вона орієнтована на автоматизацію основних бізнес-процесів СТО, надання зручного інструменту для взаємодії з клієнтами, має зручний онлайн-інтерфейс і підтримує українську мову. Серед основного функціоналу — облік клієнтів і авто, створення замовлень, графіки робіт, рахунки, склад, аналітика та автоматичні нагадування. Проте варто враховувати, що частина функцій доступна лише в розширених тарифах, а деякі можливості потребують налаштування, що може бути незручним для користувача без технічного досвіду.



Shop-Ware [5] — сучасна хмарна платформа для управління автосервісом (рис.1.4.). Вона орієнтована переважно на ринок США, але доступна і для міжнародного ринку. Попри широкий функціонал, Shop-Ware орієнтований переважно на англомовних користувачів і не має локалізації українською мовою. Крім того, підписка на сервіс є досить дорогою, що може бути бар'єром для невеликих СТО.

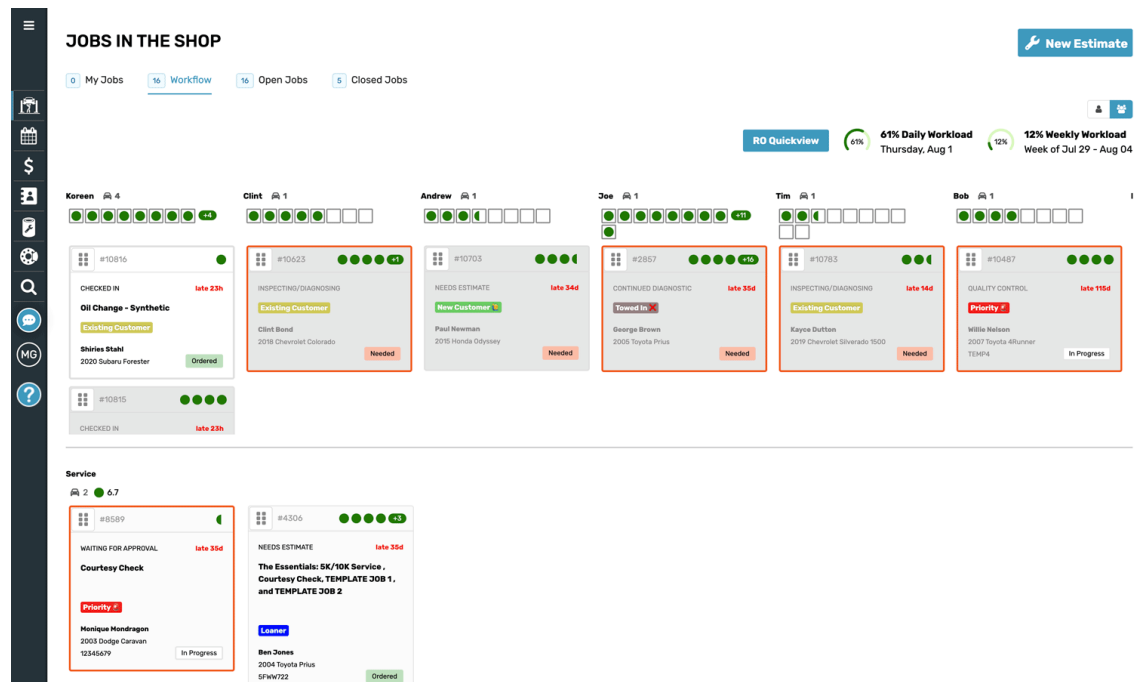


Рисунок 1.4. Інтерфейс Shop-Ware

На відміну від існуючих аналогів, розроблювана система спеціально створюється з урахуванням потреб українських СТО. Вона має низку переваг, які роблять її зручною та ефективною саме для місцевого ринку. Зокрема, інтерфейс повністю українською мовою, що полегшує використання для персоналу. Дизайн додатку простий і зрозумілий, водночас він добре працює на різних пристроях.

У системі передбачено повноцінний складський облік, можливості для ведення фінансів і автоматичне формування документів. Також реалізовано чіткий розподіл ролей користувачів, що дозволяє ефективно організувати роботу СТО: передбачені ролі власника, адміністратора, менеджера та механіка.

У таблиці 1.1 представлено порівняльний аналіз програмних рішень для СТО, таких як RemOnline, Smart CRM, Shop-Ware, AutoRepair Cloud та розроблюваний

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

додаток. Аналіз показує, що жодна з цих систем не забезпечує повного покриття потреб невеликого українського автосервісу. Часто відсутня українська локалізація, а інтерфейси можуть бути перевантаженими або складними для звичайного користувача. Крім того, обмежена кастомізація, складність у налаштуванні або висока вартість роблять деякі системи малодоступними для малого бізнесу.

Таблиця 1.1

Порівняння програмних продуктів для СТО

Характеристика/Система	RemOnline	SmartCRM	Shop-Ware	AutoRepair Cloud	Розроблюваний додаток
Тип платформи	Веб-платформа	Веб-платформа	Хмарна платформа	Хмарна платформа	Веб-платформа
Локалізація	Українська	Українська	Англійська	Англійська	Українська
Інтерфейс	Складний для новачків	Інтуїтивно зрозумілий	Професійний	Простий, функціональний	Простий
Дизайн	Сучасний	Сучасний	Сучасний, технологічний	Простий, базовий	Мінімалістичний, адаптивний
Складський облік	Так	Так	Так	Так	Так
Фінансовий модуль	Повноцінний	Базовий	Повноцінний	Присутній	Присутній
Документообіг	Повний	Шаблони, друк документів	Так	Базовий	Шаблони документів
Тип доступу	Хмарний, з особистим кабінетом	Хмарний	Хмарний	Хмарний	Авторизований через акаунт
Платність	Платна підписка	Платна підписка	Платна підписка	Платна підписка	Безкоштовний (прототип)

1.3. Вибір архітектур вебдодатку

Вибір відповідної архітектури веб-орієнтованої системи для обраної теми «Додаток для відстеження стану технічного обслуговування автомобілів» є важливим етапом у процесі розробки програмного забезпечення. Сервіс має бути ін-

туїтивно зрозумілим, доступним для широкого кола користувачів, простим у використанні та підтримці.

Основна мета архітектури є гарантування стабільності, надійності, масштабованості та керованості програмного продукту. Правильно побудована архітектура дозволяє розробникам чітко розуміти взаємозв'язки між компонентами системи, що робить розробку, тестування та майбутню підтримку продукту набагато простішим.

SPA-архітектура [8] є сучасним методом розробки веб-додатків, який широко використовується в системах, які потребують високого рівня інтерактивності. Це дозволяє розробити швидкий, зручний та адаптивний інтерфейс для кінцевого користувача без необхідності повного перезавантаження сторінок, тим самим покращуючи досвід користувачів.

Основні компоненти архітектури SPA включають:

Front-end (клієнтська частина) – це інтерфейс, який взаємодіє з користувачем у браузері. При першому завантаженні всі основні ресурси (HTML, CSS, JavaScript) завантажуються одноразово. Надалі, при переходах між сторінками або функціональними модулями, оновлюються лише необхідні частини інтерфейсу, що значно підвищує швидкодію додатку.

Back-end (серверна частина) – відповідає за логіку обробки запитів клієнта, взаємодію з базою даних, обробку замовлень, реєстрацію/авторизацію, планування замовлень, оплату тощо. Сервер повертає лише необхідні дані у форматі JSON.

API (Application Programming Interface) [9] – це набір функцій і протоколів, через які фронтенд взаємодіє з бекендом. API забезпечує стандартизований обмін даними між клієнтом і сервером. У межах SPA запити до API відправляються асинхронно (через fetch, axios тощо), і система реагує на дії користувача без оновлення сторінки.

База даних – центральне сховище інформації про користувачів, клієнтів, автомобілі, замовлення, послуги, запчастини, транзакції та документи. База даних повинна бути гнучкою, масштабованою та підтримувати зв'язки між сутностями (наприклад, автомобіль – клієнт – замовлення).

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Однією з ключових переваг сучасного веб-додатку є використання SPA-архітектури (Single Page Application), яка забезпечує швидку, плавну та зручну взаємодію користувача з інтерфейсом. Завдяки тому, що сторінка не перезавантажується повністю при кожній дії, користувач отримує майже миттєвий відгук від системи, що значно покращує загальний досвід користування.

Навігація між розділами реалізована без затримок, тому робота з додатком сприймається як робота з десктопною програмою. Крім цього, SPA дозволяє ефективно використовувати ресурси сервера, дані передаються лише в тому обсязі, який дійсно необхідний, за допомогою API-запитів. Це зменшує обсяг трафіку та пришвидшує роботу додатку.

Такий підхід також спрощує масштабування системи, адже клієнтська та серверна частини розробляються окремо. Це означає, що в майбутньому можна оновлювати або вдосконалювати одну частину, не зачіпаючи іншу. У результаті знижується навантаження на сервер і мережу, що позитивно впливає на продуктивність та стабільність роботи всього застосунку.

Таким чином, архітектура SPA з REST API є оптимальним вибором для веб-додатку для відстеження стану ТО автомобілів. Вона забезпечує необхідну швидкість, зручність, масштабованість та відповідає сучасним вимогам до веб-додатків.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Після всебічного аналізу завдання та дослідження аналогічних інформаційних систем, наступним логічним кроком є визначення інструментальних засобів реалізації програмного комплексу та постановка вимог до нього.

Для створення серверної частини системи обрано мову програмування C# у поєднанні з платформою .NET. Такий вибір обумовлений рядом важливих переваг. Насамперед, C# є сучасною мовою з високим рівнем типобезпеки та доброю підтримкою об'єктно-орієнтованого підходу, що дозволяє реалізовувати складну логіку в структурованому та надійному вигляді.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Платформа .NET [11] забезпечує широкі можливості для розробки: вона включає розвинений набір вбудованих бібліотек, інструментів для роботи з базами даних, веб-сервісами, автентифікацією тощо. Це значно прискорює процес створення програмного продукту та спрощує його подальшу підтримку.

Окрім того, екосистема .NET активно підтримується компанією Microsoft, має потужну документаційну базу та велику спільноту розробників. Це дозволяє вирішувати нетривіальні задачі за допомогою перевірених практик і широко доступних рішень, що робить її ефективним вибором для реалізації проєктів середнього та великого масштабу.

Клієнтська частина додатку реалізовуватиметься з використанням React[6] – популярної JavaScript-бібліотеки для побудови інтерфейсів користувача. React дозволяє створювати динамічні односторінкові застосунки (SPA), які забезпечують швидке оновлення даних на сторінці без її повного перезавантаження. Компонентний підхід у React дозволяє розробляти повторно використовувані та добре структуровані частини інтерфейсу, що значно спрощує підтримку та розвиток системи.

Також у розробці клієнтської частини планується використання бібліотеки компонентів для React (Ant Design [12]). Така бібліотека надає готові до використання інтерфейсні елементи, стилізовані згідно з принципами сучасного дизайну. Це дозволяє значно пришвидшити розробку інтерфейсу, забезпечити візуальну узгодженість між різними частинами системи та зменшити кількість помилок при верстці. Крім того, використання компонентної бібліотеки покращує доступність та адаптивність інтерфейсу під різні розміри екранів і пристроїв.

Для ефективного запуску та розробки клієнтської частини використовується Vite [14] – сучасний інструмент для збирання фронтенду, який забезпечує надзвичайно швидкий старт розробки завдяки попередній компіляції залежностей та використанню нативних можливостей браузеру. Vite забезпечує гаряче оновлення модулів (HMR), мінімальний час запуску локального сервера та високу продуктивність при збиранні проєкту, що суттєво прискорює цикл розробки й тестування інтерфейсу.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Серверна частина системи буде створена за допомогою платформи .NET з використанням мови програмування C#. Сучасна платформа .NET є кросплатформною та об'єднує можливості попередніх рішень .NET Framework [15] та .NET Core [11]. Вона забезпечує високу продуктивність, безпеку, а також має широку підтримку з боку спільноти та корпорації Microsoft, що надає переваги у вигляді широкої документації та доступних інструментів розробки.

Для роботи з базою даних застосовується Entity Framework Core [16] — ORM (Object-Relational Mapping) фреймворк, що дозволяє взаємодіяти з реляційною базою даних, використовуючи об'єктно-орієнтований підхід. Застосування підходу Code First [18] дає змогу визначати структуру бази даних безпосередньо у коді, а не вручну створювати таблиці, що спрощує процес розробки та знижує ризик помилок.

У якості системи управління базами даних обрано Microsoft SQL Server (MS SQL)[19]. Ця СУБД добре інтегрується з .NET-платформою та забезпечує високу продуктивність, надійність і підтримку складних SQL-запитів. Завдяки потужному функціоналу MS SQL можливо ефективно реалізувати зберігання, фільтрацію та аналіз великого обсягу даних, пов'язаних з розкладом сеансів, інформацією про фільми, користувачів та замовлення.

Такий набір технологій дозволяє створити сучасну, гнучку та масштабовану систему, що відповідає актуальним вимогам до веб-додатків. Крім того, чітке розділення обов'язків між клієнтською та серверною частинами спрощує тестування, супровід та подальший розвиток програмного продукту.

Висновки до розділу 1

У межах першого розділу було послідовно розглянуто всі основні етапи, необхідні для розробки сучасного веб-додатку, орієнтованого на автоматизацію робочих процесів у сфері технічного обслуговування. Визначено мету проєкту та обґрунтовано його актуальність.

Далі проаналізовано існуючі рішення на ринку, що дозволило сформулювати бачення функціональності та особливостей майбутнього продукту з урахуванням

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

переваг і недоліків аналогів. Вибір архітектури системи базувався на потребі забезпечити зручний користувацький досвід, тому було обрано на SPA-підхід, який дозволяє створити швидкий та адаптивний інтерфейс.

Визначено доцільність використання .NET та мови програмування C# для серверної частини, а також бібліотеки React для клієнтського інтерфейсу. Обрано інструменти, що сприяють швидкій розробці та подальшому масштабуванню.

Загалом проведений аналіз і розроблена концепція створюють надійну основу для реалізації ефективного цифрового продукту, здатного підвищити якість і продуктивність роботи СТО.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перед розробкою програмного забезпечення для автоматизації діяльності станції технічного обслуговування автомобілів (СТО) було проведено попередній аналіз вимог до майбутньої системи. Основна мета створення даного програмного комплексу полягає в автоматизації основних бізнес-процесів СТО, таких як ведення клієнтської бази, облік замовлень на послуги, контроль виконання робіт, управління персоналом та запасами, а також формування звітності. Відповідно, під час проєктування враховувалися вимоги різних типів користувачів, що мають взаємодіяти з системою, а також визначалися загальні сценарії її використання.

У рамках функціональних вимог було визначено основні ролі користувачів, серед яких: власник, адміністратор системи, менеджер замовлень, механік (виконавець робіт).

На рисунку 2.1. представлено варіанти використання системи у вигляді діаграми варіантів використання (Use Case Diagram [20]), яка дозволяє наочно побачити основні дії, які можуть виконувати користувачі різних ролей. Така діаграма є частиною мовного стандарту UML і широко використовується при проєктуванні програмного забезпечення для моделювання взаємодії між акторами (користувачами) та системою.

На діаграмі використання зображено чотири основні типи користувачів: власник, адміністратор, менеджер і механік. Кожен із них має власний набір можливих дій, які доступні після авторизації в системі.

Власник контролює ключові процеси на СТО. Він має змогу створювати та редагувати замовлення, переглядати інформацію про клієнтів, працівників і ресурси, управляти товарами на складі та послугами, слідкувати за транзакціями. Власник може вносити зміни до інформації про компанію, переглядати розклад у ка-

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

лендарі, а також генерувати QR-коди для оплати, що дозволяє автоматизувати й оптимізувати фінансові процеси.

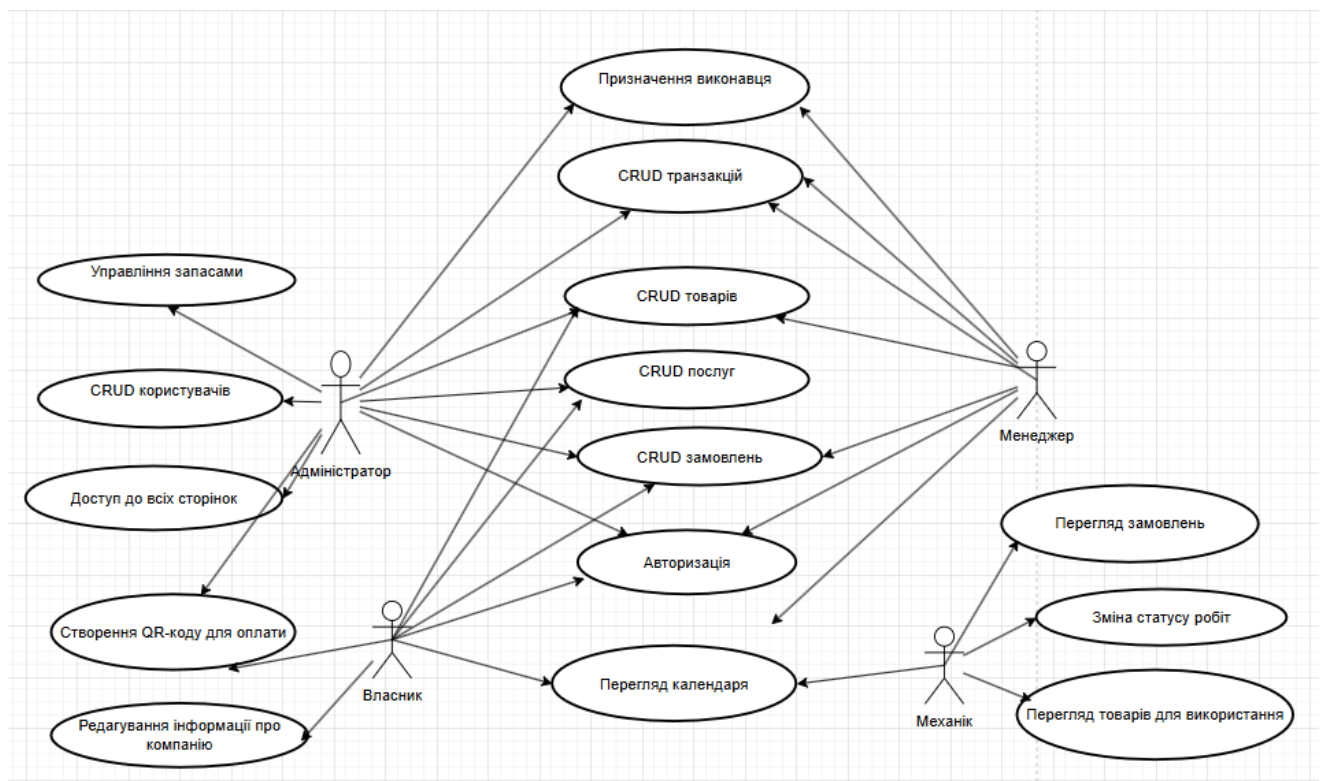


Рисунок 2.1. Діаграма варіантів використання

Адміністратор виконує функції повного управління системою. Він має доступ до всіх сторінок і може здійснювати повний CRUD (створення, редагування, перегляд, видалення) для всіх сутностей: замовлень, клієнтів, користувачів, товарів, послуг, ресурсів, транзакцій та закупівель. Адміністратор також може управляти складом, що дає змогу здійснювати централізований контроль над роботою сервісу. Його роль передбачає максимальні повноваження для підтримки стабільності та ефективності функціонування системи.

Менеджер відповідає за операційне керування замовленнями. Він створює та редагує заявки на ремонт або обслуговування, вносить дані про клієнтів, обирає необхідні послуги й ресурси, додає товари зі складу до замовлень. Менеджер не має змоги змінювати транзакції, але може їх переглядати. Його роль зосереджена на взаємодії з клієнтами, контролі виконання послуг і забезпеченні комунікації між технічним та адміністративним персоналом.

Механік має доступ лише до тих функцій, що безпосередньо стосуються виконання його технічних завдань. Він бачить тільки власні замовлення, може змінювати статус виконаних робіт (наприклад, від "у роботі" до "завершено"), переглядати інформацію про використані товари та ознайомлюватися з деталями клієнта. Додатково механік має змогу переглядати розклад у календарі, щоб знати про майбутні завдання. Таким чином, його інтерфейс є максимально простим і орієнтованим на продуктивність.

Нефункціональні вимоги [26] до системи є не менш важливими, ніж функціональні, адже саме вони визначають якість роботи програмного продукту, його зручність у використанні, безпеку та ефективність. У межах розробки веб-орієнтованої системи для відстеження ТО автомобілів було сформовано низку ключових нефункціональних вимог, які враховувались під час проєктування та реалізації програмного комплексу.

Перш за все, система повинна відповідати високим вимогам до продуктивності. Це означає, що час реакції на дії користувача має бути мінімальним. За базового навантаження (до кількох одночасних користувачів) очікується, що основні операції, такі як перегляд списку замовлень, зміна статусу, створення клієнта або генерація документа, виконуватимуться за 2-7 секунди. Така швидкість відповіді забезпечує комфортну взаємодію та запобігає виникненню затримок у процесі обслуговування клієнтів на СТО.

Крім цього, надзвичайно важливою є надійність системи. Усі дані, що зберігаються у базі даних, мають бути захищеними від втрат та помилок. Передбачено реалізацію механізмів автоматичного збереження змін, а також резервного копіювання на випадок збоїв або втрати доступу до основного серверу. Надійність також включає безпомилкову роботу всіх модулів системи у типових ситуаціях експлуатації.

Не менш важливою є безпека даних. Оскільки система оперує персональними та технічними даними клієнтів, а також фінансовою інформацією, необхідно впровадити надійні засоби контролю доступу. Передбачено розмежування прав доступу залежно від ролі користувача в системі (власник, адміністратор, мене-

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

джер, механік). Наприклад, механіки не матимуть доступу до персональних даних клієнтів, а менеджери не зможуть змінювати налаштування компанії. Крім того, реалізовано механізми аутентифікації користувачів та захищене зберігання паролів у хешованому вигляді.

Ще однією важливою характеристикою є зручність у використанні. Інтерфейс користувача має бути інтуїтивно зрозумілим навіть для людей, які не мають спеціальної технічної підготовки. Навігація між екранами, створення нових записів, перегляд історії та звітів повинні здійснюватися максимально просто. Для цього застосовуються сучасні підходи до UI/UX-дизайну, включаючи адаптивність інтерфейсу для використання на різних пристроях — від комп'ютера до планшета чи смартфона.

У сукупності всі ці нефункціональні вимоги спрямовані на створення надійного, зручного та безпечного програмного продукту, який буде ефективно працювати в умовах реального бізнес-середовища СТО.

Під час розробки програмного забезпечення було застосовано підхід Code First, який є частиною технології Entity Framework Core. Цей підхід передбачає створення структури бази даних безпосередньо з коду, що дозволяє проектувати модель даних у вигляді C#-класів, а вже на їх основі — автоматично генерувати схему бази даних.

З метою наочного відображення структури системи було створено діаграму сутностей (Рис. 2.2). Перед її побудовою варто визначити, що саме вона ілюструє.

Діаграма сутностей (ERD) [22] – це графічне представлення логічної структури бази даних, яке відображає основні сутності системи, їхні атрибути та зв'язки між ними. Така діаграма дозволяє наочно описати інформаційні об'єкти, з якими працює система, а також визначити правила їхньої взаємодії. Використовується на етапі проектування бази даних для забезпечення чіткої структуризації даних та їхньої подальшої реалізації у вигляді таблиць у СКБД.

Усі сутності в системі успадковуються від базової сутності BaseEntity. Ця сутність містить єдину властивість Id типу Guid, що виступає універсальним ідентифікатором для всіх об'єктів у базі даних. Завдяки цьому забезпечується уніфі-

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

кована структура ідентифікації сутностей, що спрощує роботу з ORM та реалізацію CRUD-операцій.

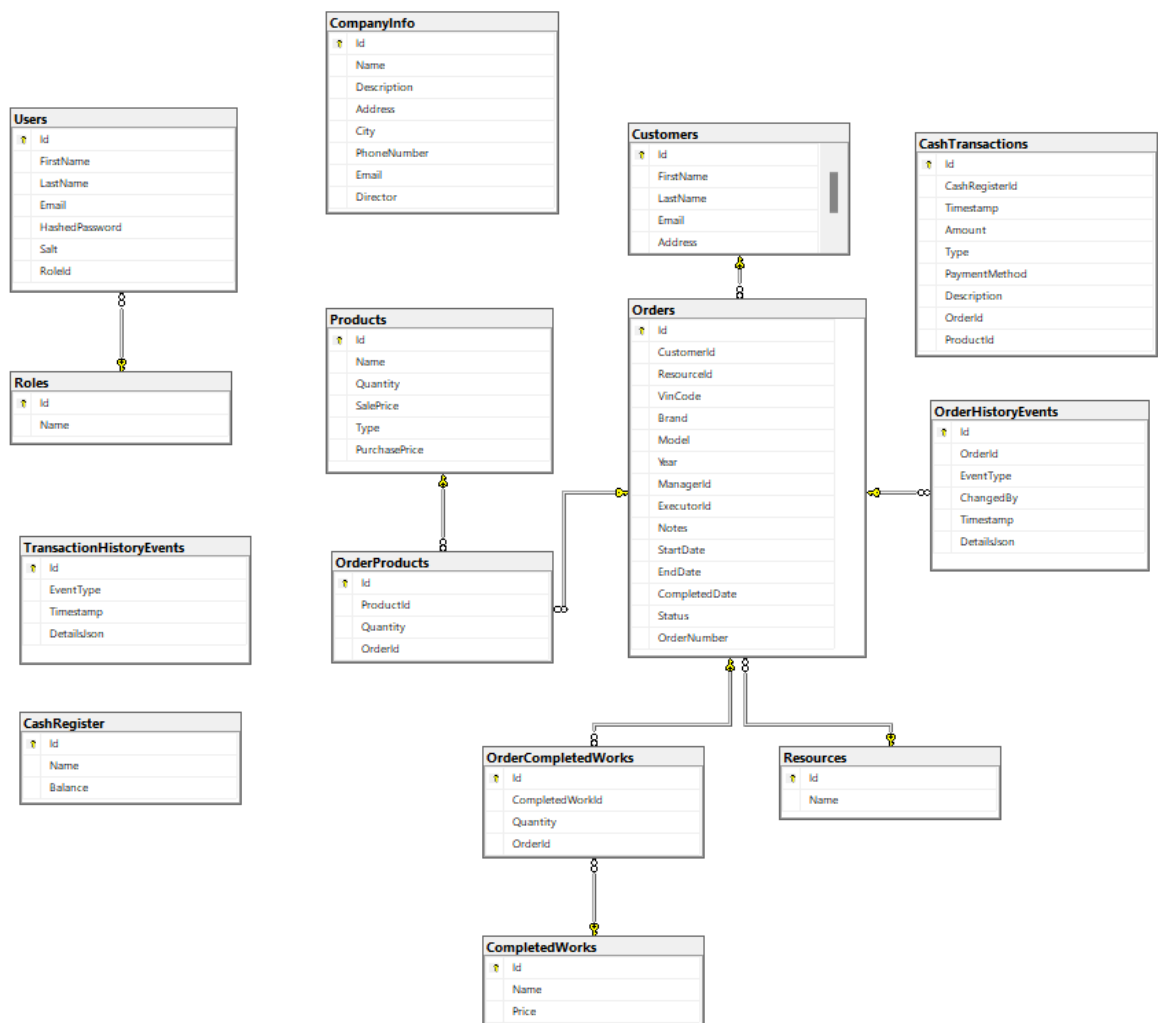


Рисунок 2.2. Діаграма сутностей бази даних системи

Ключовою сутністю системи є сутність Order, вона відображає одне замовлення на СТО. Вона пов'язана із клієнтом і ресурсом, містить інформацію про автомобіль (VIN-код, марку, модель, рік випуску), менеджера, виконавця (механіка), статус замовлення, додаткові примітки, а також дати початку, закінчення і завершення робіт. Крім того, замовлення має списки виконаних робіт та використаних товарів, що дозволяє відслідковувати повну історію обслуговування та використаних ресурсів.

Сутність Product зберігає інформацію про товар або матеріал, що є на складі СТО. Вона включає назву товару, кількість одиниць, ціну закупки і ціну продажу,

а також тип товару. Завдяки цій сутності реалізується функціональність складського обліку та продажу товарів.

Сутність `CompletedWork` представляє одну конкретну послугу або роботу, що може бути виконана в межах замовлення. Вона містить назву послуги та її вартість. Сутність має зв'язок із таблицею зв'язку `OrderCompletedWork`, що дозволяє відобразити, які саме роботи були виконані в межах конкретного замовлення.

Сутність `OrderCompletedWork` – службова сутність для зв'язку між замовленням і виконаними роботами. Містить кількість виконаних робіт, ідентифікатори замовлення та роботи. Пов'язана з сутностями `Order` і `CompletedWork`.

Сутність `OrderProduct` відображає використання товарів у конкретному замовленні. Містить кількість, ідентифікатори замовлення та товару. Пов'язана із сутностями `Order` і `Product`.

Сутність `Customer` зберігає дані клієнта (ім'я, прізвище, email та адресу). Пов'язується із замовленнями через `Order`.

Сутність `CompanyInfo` містить загальну інформацію про компанію (назву, опис, адресу, місто, телефон, email і директора). Використовується для відображення або редагування реквізитів компанії.

Сутність `OrderHistoryEvent` зберігає історію змін, пов'язаних із замовленням. Містить тип події, дату, автора змін та JSON з деталями. Пов'язана з сутністю `Order`.

Сутність `Resource` представляє ресурс, який може бути закріплений за замовленням. Містить лише назву.

Сутність `Role` описує роль користувача в системі (адміністратор, менеджер, механік). Містить назву ролі.

Сутність `User` містить облікові дані користувача – ім'я, прізвище, email, хеш пароля, сіль та роль. Пов'язана з сутністю `Role`.

Сутність `CashRegister` відображає касу СТО з назвою та поточним балансом. Може бути джерелом або одержувачем грошових операцій.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Сутність CashTransaction представляє фінансову операцію. Містить суму, тип транзакції, спосіб оплати, дату, опис, а також зв'язки з касою, замовленням або продуктом.

Сутність TransactionHistoryEvent фіксує історію змін, пов'язаних із фінансовими транзакціями. Містить тип події, дату і JSON з деталями.

Кожна сутність у системі має набір властивих їй методів, що реалізують ключові операції, зокрема створення, редагування, видалення та перегляд записів. Ці дії є типовими для більшості об'єктів у системі. Водночас деякі сутності містять функціональність, характерну лише для них. Наприклад, об'єкт «Order» реалізує зміну статусу виконання, що є специфічною дією в контексті управління замовленнями.

Побудова діаграми сутностей дозволила структурувати логіку взаємодії між елементами системи та заклала основу для реалізації об'єктно-орієнтованого підходу в розробці програмного забезпечення.

2.2. Розробка бази даних системи

Для зберігання й обробки даних у програмному комплексі було використано Microsoft SQL Server (MS SQL). MS SQL — це потужна реляційна система керування базами даних, розроблена компанією Microsoft. Вона забезпечує надійне зберігання, ефективний доступ до даних і високу продуктивність у великих корпоративних середовищах.

MS SQL відзначається розвиненим функціоналом для роботи з транзакціями, підтримкою складних запитів, процедур і тригерів, а також гнучкими засобами контролю доступу. Однією з ключових переваг цієї СУБД є її тісна інтеграція з іншими продуктами Microsoft, зокрема середовищем .NET, що робить її ідеальним вибором для створення сучасних клієнт-серверних і веб-орієнтованих додатків.

Також варто зазначити, що MS SQL підтримує масштабування, резервне копіювання, засоби реплікації та моніторингу продуктивності, що дозволяє ефективно працювати з великими обсягами даних і гарантує надійність та безперервність

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

роботи програмного забезпечення. Завдяки цьому MS SQL є популярним рішенням у сфері розробки прикладних систем, зокрема у проєктах, пов'язаних з автоматизацією роботи СТО.

У рамках даного програмного комплексу було створено набір таблиць, які відображають основні сутності системи. Нижче подано детальний опис структури кожної таблиці, її полів, типів даних, а також первинних (РК) та зовнішніх (FK) ключів.

Таблиця orders відповідає за облік замовлень, що створюються у системі, табл. 2.1.

Таблиця 2.1.

Структура таблиці orders

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор замовлення
customer_id	uniqueidentifier	–	+	Зовнішній ключ на клієнта
resource_id	uniqueidentifier	–	+	Зовнішній ключ на ресурс
vin_code	varchar(50)	–	–	VIN-код автомобіля
brand	varchar(50)	–	–	Марка автомобіля
model	varchar(50)	–	–	Модель автомобіля
year	int	–	–	Рік випуску
manager_id	uniqueidentifier	–	–	Ідентифікатор менеджера
executor_id	uniqueidentifier	–	–	Ідентифікатор виконавця
status	int	–	–	Статус замовлення
notes	text	–	–	Примітки до замовлення
start_date	datetimeoffset	–	–	Дата початку робіт
end_date	datetimeoffset	–	–	Очікувана дата завершення
completed_date	datetime	–	–	Фактична дата завершення

Для збереження інформації про продукти, створено таблицю product, табл. 2.2.

Таблиця 2.2.

Структура таблиці product

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор товару
name	varchar(100)	–	–	Назва товару

Продовження таблиці 2.2

quantity	int	—	—	Кількість на складі
purchase_price	decimal(10,2)	—	—	Ціна закупівлі
sale_price	decimal(10,2)	—	—	Ціна продажу
type	int	—	—	Тип товару

Таблиця completedWorks зберігає дані про послуги, що надаються СТО, табл. 2.3.

Таблиця 2.3.

Структура таблиці completedWorks

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор роботи
name	varchar(100)	—	—	Назва виконаної роботи
price	decimal(10,2)	—	—	Вартість виконаної роботи

Таблиця orderCompletedWork службова сутність для зв'язку між замовленням і виконаними роботами табл. 2.4. Містить кількість виконаних робіт, ідентифікатори замовлення та роботи

Таблиця 2.4.

Структура таблиці orderCompletedWork

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор запису
completed_work_id	uniqueidentifier	—	+	Зовнішній ключ на CompletedWork
quantity	decimal(10,2)	—	—	Кількість виконаних робіт
order_id	uniqueidentifier	—	+	Зовнішній ключ на Order

Для збереження інформації про клієнтів, створено таблицю customer, табл. 2.5. Вона містить персональні та контактні дані клієнта, а також ідентифікатор для зв'язку з іншими таблицями.

Таблиця 2.5.

Структура таблиці customer

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор

Продовження таблиці 2.5.

quantity	int	—	—	Кількість на складі
purchase_price	decimal(10,2)	—	—	Ціна закупівлі
sale_price	decimal(10,2)	—	—	Ціна продажу
type	int	—	—	Тип товару

Таблиця resource зберігає назву ресурса, який може бути закріплений за замовленням, табл. 2.6. Містить лише назву та унікальний ідентифікатор ресурсу.

Таблиця 2.6.

Структура таблиці resource

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор ресурсу
name	varchar(100)	—	—	Назва ресурсу

Таблиця user зберігає облікові дані користувача, табл. 2.7. Пов'язана з таблицею role.

Таблиця 2.7.

Структура таблиці user

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор користувача
first_name	varchar(128)	—	—	Ім'я користувача
last_name	varchar(128)	—	—	Прізвище користувача
email	varchar(128)	—	—	Електронна пошта
hashed_password	varchar(128)	—	—	Хеш паролю
salt	varchar(128)	—	—	Сіль для хешування паролю
role_id	uniqueidentifier	—	+	Зовнішній ключ на роль

Таблиця cashRegister зберігає назву та баланс каси СТО, табл. 2.8.

Таблиця 2.8.

Структура таблиці cashRegister

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	—	Унікальний ідентифікатор каси
name	varchar(100)	—	—	Назва касового апарата
balance	decimal(10,2)	—	—	Поточний баланс каси

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Таблиця role зберігає роль користувача в системі, табл. 2.9.

Таблиця 2.9.

Структура таблиці role

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор
name	varchar(50)	–	–	Назва ролі

Таблиця cashTransaction зберігає фінансову операцію, табл. 2.10.

Таблиця 2.10.

Структура таблиці cashTransaction

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор транзакції
cash_register_id	uniqueidentifier	–	+	Зовнішній ключ на касовий апарат
timestamp	datetime	–	–	Дата й час транзакції
amount	decimal(10,2)	–	–	Сума
type	int	–	–	Тип транзакції (прибуток/витрати)
payment_method	int	–	–	Метод оплати (готівка, картка)
description	text	–	–	Опис транзакції
order_id	uniqueidentifier	–	+	Зовнішній ключ на замовлення (опціонально)
product_id	uniqueidentifier	–	+	Зовнішній ключ на товар (опціонально)

Таблиця transactionHistoryEvent зберігає фіксує історію змін, пов'язаних із фінансовими транзакціями, табл. 2.11.

Таблиця 2.11.

Структура таблиці transactionHistoryEvent

Назва	Тип даних	РК	FK	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор події
event_type	varchar(100)	–	–	Тип події
timestamp	datetime	–	–	Час події
details_json	text	–	–	JSON-інформація про зміни

Таблиця orderHistoryEvent зберігає історію змін, пов'язаних із замовленням, табл. 2.12.

Таблиця 2.12.

Структура таблиці orderHistoryEvent

Назва	Тип даних	РК	ФК	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор події
order_id	uniqueidentifier	–	+	Зовнішній ключ на Order
event_type	varchar(100)	–	–	Тип події
changed_by	varchar(100)	–	–	Ким була змінена подія
timestamp	datetime	–	–	Час події
details_json	text	–	–	JSON-деталі змін

Таблиця companyInfo зберігає загальну інформацію про компанію, табл. 2.13.

Таблиця 2.13.

Структура таблиці companyInfo

Назва	Тип даних	РК	ФК	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор компанії
name	varchar(100)	–	–	Назва компанії
description	text	–	–	Опис компанії
address	varchar(255)	–	–	Адреса
city	varchar(100)	–	–	Місто
phone_number	varchar(20)	–	–	Номер телефону
email	varchar(100)	–	–	Електронна пошта
director	varchar(100)	–	–	ПІБ директора

Таблиця orderProduct зберігає використання товарів у конкретному замовленні, табл. 2.14.

Таблиця 2.14.

Структура таблиці orderProduct

Назва	Тип даних	РК	ФК	Опис поля
id	uniqueidentifier	+	–	Унікальний ідентифікатор запису
product_id	uniqueidentifier	–	+	Зовнішній ключ на Product
quantity	decimal(10,2)	–	–	Кількість використаного товару
order_id	uniqueidentifier	–	+	Зовнішній ключ на Order

Як результат, розроблена база даних забезпечує ефективне зберігання, зв'язок та обробку інформації, необхідної для функціонування системи автоматизації СТО. Всі сутності мають чітко визначені зв'язки, що дозволяє легко масштабувати систему та розширювати її функціональні можливості в майбутньому.

2.3. Проєктування та реалізація алгоритмів роботи системи

У процесі розробки веб-орієнтованого програмного комплексу для автоматизації діяльності СТО важливою складовою є проєктування логіки взаємодії між основними елементами системи. Ця взаємодія реалізується у вигляді алгоритмів, що визначають порядок обробки даних, зміну їх станів та передачу між окремими модулями й компонентами.

Алгоритми роботи системи реалізують типові сценарії, з якими стикаються користувачі — від створення замовлення та призначення відповідального виконавця, до формування документів і завершення виконаних робіт. Основна логіка програми зосереджена на контролі правильності вводу даних, перевірці зв'язків між сутностями та актуалізації статусів у відповідності до дій користувача.

Сценарії використання системи охоплюють типові дії, які будуть виконуватись у процесі її експлуатації. Наприклад, при надходженні нового клієнта, менеджер створює в системі новий профіль замовника, вносить необхідні дані про автомобіль, тип необхідного обслуговування та обрані послуги. Далі система дозволяє призначити відповідального механіка та створити замовлення, яке зберігається у базі даних зі статусом "Нове". У міру виконання робіт механік оновлює статус свого завдання, і після завершення замовлення менеджер має змогу змінити його статус на "Завершено", з можливістю формування акта виконаних робіт або рахунку для клієнта.

Окрім описаних сценаріїв, система також підтримує можливість редагування вже створених замовлень, пошук інформації за різними критеріями (ПІБ клієнта, державний номер автомобіля, дата виконання робіт тощо).

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Діаграма діяльності є одним із ключових інструментів для зображення динаміки роботи системи. Вона нагадує вдосконалену блок-схему та дозволяє наочно показати, як відбувається перехід від однієї дії до іншої. На рисунку 2.3. представлено приклад такої діаграми, що ілюструє послідовність операцій у межах функціонування системи.

На діаграмі діяльності (Рис.2.3) проілюстровано основні дії адміністратора менеджера та механіка під час роботи з системою, демонструючи логіку їхньої взаємодії з інтерфейсом та між собою. Вона починається з точки входу в систему, яка є спільною для усіх ролей.

Процес розпочинається з дії Менеджера – вхід у систему. Спочатку відбувається перевірка облікових даних, після чого система здійснює перевірку наявності авторизації. Якщо авторизація пройдена успішно, користувач потрапляє до головного меню. У випадку помилки доступ до системи припиняється, і відбувається перехід до повідомлення про помилку входу.

Для Менеджера в головному меню доступна функція редагування даних обслуговування. Цей процес передбачає кілька дій: спочатку система здійснює фільтрацію або сортування доступних записів, після чого менеджер може обрати конкретне замовлення для редагування. У середині цього процесу можливе редагування даних, додавання робіт та товарів, а також призначення виконавця. Після цього відбувається збереження змін, і менеджер може вийти із системи.

Окрім редагування замовлень, Менеджеру доступний також перегляд графіка обслуговування та реалізована функція додавання транзакцій. Цей процес передбачає введення необхідних даних про оплату або витрати і подальше збереження цієї інформації у базі.

Адміністратор також починає роботу з системою із входу, після чого отримує доступ до головного меню. У ньому він має змогу додавати нові записи обслуговування, вводити відповідні дані, а також керувати працівниками. Як і у випадку з Менеджером, Адміністратор має доступ до перегляду графіка обслуговування, а також, у разі потреби, виходить із системи. Якщо ж відбувається помилка

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

під час входу або на будь-якому етапі перевірки, користувач перенаправляється до повідомлення про помилку входу.

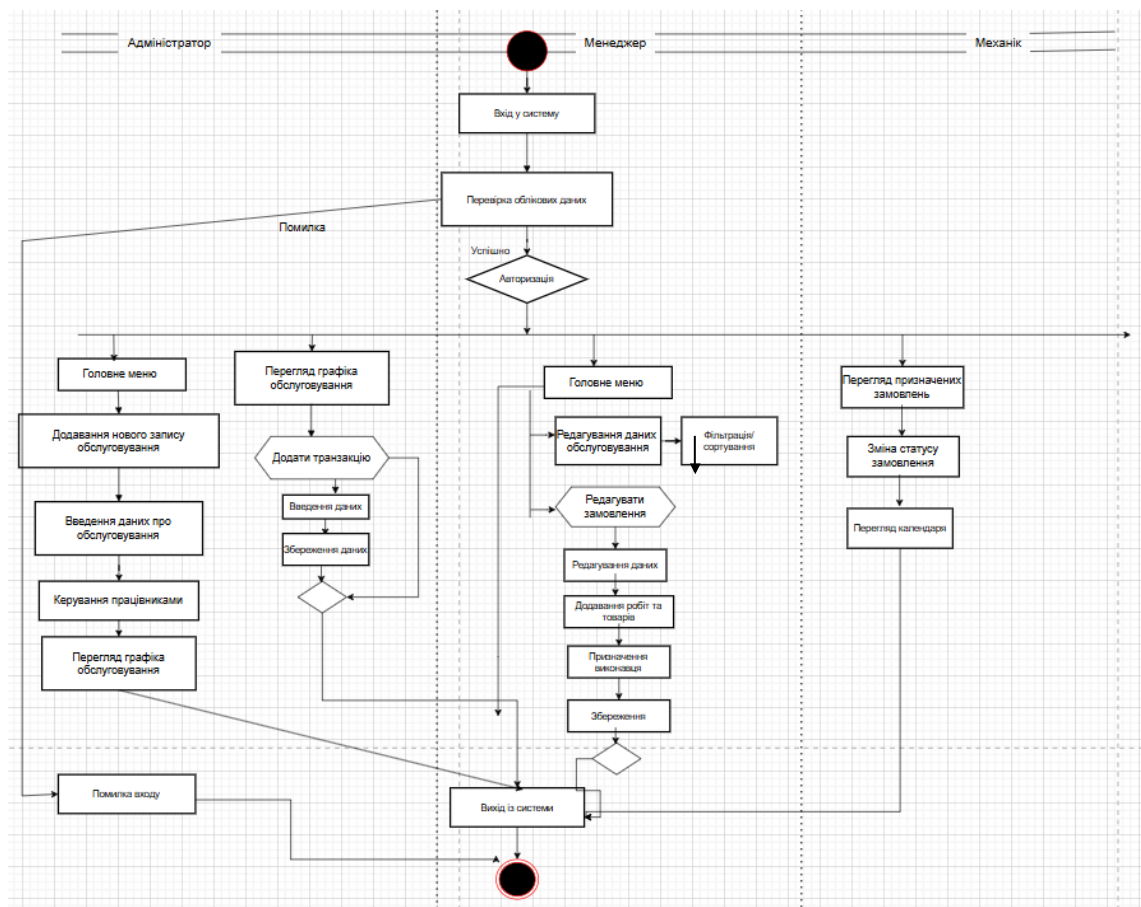


Рисунок 2.3. Діаграма діяльності

Окрему частину діаграми займає діяльність Механіка. Його функціонал є більш вузьким і зосереджений на виконанні призначених йому замовлень. Механік має можливість переглядати ті замовлення, які були на нього призначені Менеджером, а також змінювати статус цих замовлень у відповідності до етапів виконання. Після перегляду календаря або оновлення статусу роботи, він також може вийти із системи.

Уся діаграма побудована лінійно з умовними розгалуженнями у вигляді ромбів, які відповідають за перевірки або вибір подальшої дії. Процеси логічно завершуються точкою виходу із системи, яка є спільною для всіх ролей. Таким чином, діаграма показує типові сценарії роботи з програмним забезпеченням СТО, охоплюючи введення, обробку, редагування, призначення виконавців, фінансову частину та зміну статусу виконання робіт.

Далі розглянемо діаграму послідовності процесу створення замовлення в системі управління автосервісом (Рис.2.4). Даграма демонструє типову взаємодію між основними компонентами програмного комплексу. На початку сценарію менеджер, тобто користувач системи, відкриває відповідну форму в інтерфейсі користувача. Інтерфейс надсилає запит на ініціалізацію форми до контролера, після чого контролер звертається до логіки обробки замовлень з метою отримання необхідних даних для заповнення — зокрема, списку клієнтів, інформації про автомобілі, доступні послуги тощо.

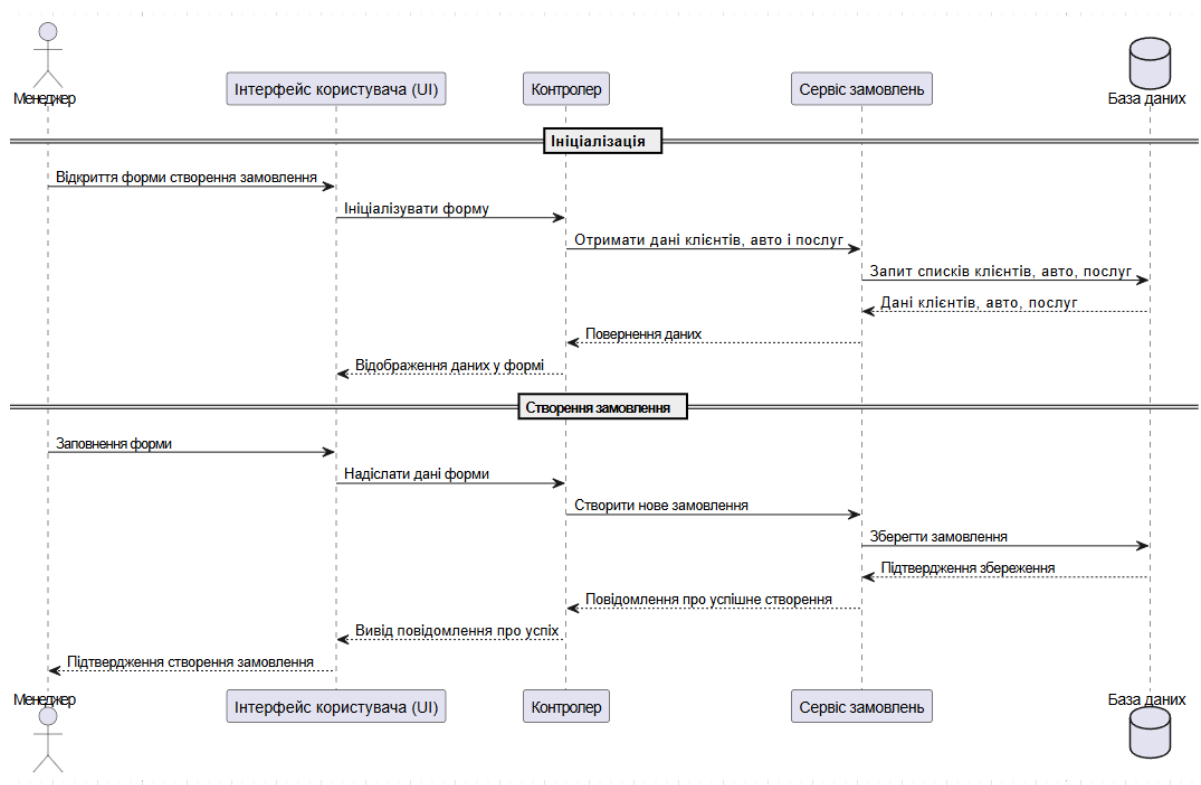


Рисунок 2.4. Діаграма послідовності

Сервіс замовлень звертається до бази даних, щоб отримати актуальну інформацію, яка потім передається назад через контролер до інтерфейсу користувача. Таким чином, менеджеру відображається готова до заповнення форма, що вже містить відповідні вибірки та підказки для швидкого створення замовлення.

Після заповнення форми менеджером інтерфейс користувача передає всі введені дані назад у контролер. Контролер, у свою чергу, передає ці дані у сервіс замовлень для обробки та збереження. OrderService записує нове замовлення до

бази даних, після чого база даних повертає підтвердження про успішне збереження.

Це підтвердження передається далі через ті ж компоненти у зворотному напрямку: від OrderService до контролера, від контролера до інтерфейсу користувача. В результаті, менеджер бачить повідомлення про те, що замовлення успішно створено. Такий ланцюжок дій забезпечує чітку та послідовну логіку роботи системи, дозволяючи зручно управляти усіма процесами, пов'язаними з обслуговуванням клієнтів.

2.4. Реалізація додатку для відстеження технічного обслуговування автомобілів

Реалізація програмного комплексу здійснювалась із застосуванням сучасних технологій та фреймворків, які забезпечують гнучкість, масштабованість і зручність розробки. Для створення серверної частини використовувався фреймворк .NET, зокрема ASP.NET Core. Клієнтську частину реалізовано на React, з використанням Vite. Для зберігання даних була обрана MS SQL Server, реляційна система управління базами даних.

Проект реалізовано з урахуванням принципів Clean Architecture, що дозволяє забезпечити чіткий поділ відповідальностей між компонентами системи, полегшити тестування та масштабування. Архітектура проєкту включає кілька основних шарів:

GraduationWork.Domain містить основні сутності, доменні моделі та інтерфейси, які не залежать від реалізації. Тут визначено сутності, такі як Order, User, Customer, Product, Resource, CompanyInfo а також відповідні перелічення, типи та базову логіку.

GraduationWork.Application містить бізнес-логіку системи, описану у вигляді сервісів, команд і запитів за патерном CQRS із використанням MediatR. У цьому шарі реалізовано обробку запитів на створення та редагування замовлень, взаємодію з клієнтами, зміну статусів, облік послуг тощо.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

GraduationWork.Infrastructure відповідає за взаємодію з зовнішніми ресурсами, зокрема з базою даних Microsoft SQL Server за допомогою Entity Framework Core. У цьому шарі реалізовано репозиторії, сервіси для роботи з файлами (наприклад, PDF-звітами), логування та інтеграції.

GraduationWork.WebApi – вхідна точка системи, яка реалізує контролери, обробляє HTTP-запити, забезпечує автентифікацію, авторизацію, валідацію вхідних даних та інші веб-залежності. Контролери отримують запити від клієнтського інтерфейсу (React) і делегують обробку до відповідних MediatR-хендлерів, реалізованих у шарі Application.

Розглянемо один із ключових елементів реалізації серверної частини — клас ApplicationDbContext, який знаходиться у проєкті GraduationWork.Infrastructure. Цей клас є основним інтерфейсом взаємодії з базою даних і наслідує від DbContext – стандартного компонента Entity Framework Core. Саме через нього здійснюється доступ до таблиць у базі, виконання запитів, збереження змін тощо. Серед публічних властивостей класу бачимо набір DbSet<T>, які відповідають таблицям у базі даних.

```
public DbSet<User> Users { get; set; }  
public DbSet<Role> Roles { get; set; }  
public DbSet<Product> Products { get; set; }  
...
```

Також у методі OnModelCreating реалізовано кастомне перетворення переліку (enum) ProductType в числове значення для зберігання в БД:

```
modelBuilder.Entity<Product>()  
    .Property(s => s.Type)  
    .HasConversion<int>();
```

Це дозволяє зручно зберігати типи продуктів у вигляді чисел, зберігаючи можливість працювати з переліками у коді.

Далі розглянемо реалізацію двох важливих сервісів, які відповідають за безпеку системи – PasswordService (Додаток А) та TokenService (Додаток Б). Їхня функціональність охоплює захист паролів користувачів та генерацію токенів для авторизації.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Почнемо з PasswordService. У цьому класі реалізовано дві основні функції – хешування пароля та його перевірка. Хешування реалізується з використанням алгоритму PBKDF2 (Rfc2898DeriveBytes). Для кожного пароля генерується унікальна сіль (salt) за допомогою RandomNumberGenerator.GetBytes. Потім пароль і сіль хешуються з великою кількістю ітерацій (у нашому випадку — 350 000), що додатково ускладнює злом. Метод повертає пару: хешований пароль у шістнадцятковому форматі та сіль у base64.

```
public (string hashedPassword, string salt) HashPassword(string password)
{
    var salt = RandomNumberGenerator.GetBytes(KeySize);
    var hash = Rfc2898DeriveBytes.Pbkdf2(
        Encoding.UTF8.GetBytes(password),
        salt,
        Iterations,
        _hashAlgorithm,
        KeySize);
    return (Convert.ToHexString(hash), Convert.ToBase64String(salt));
}
```

Метод VerifyPassword виконує зворотню операцію – перевіряє, чи співпадає наданий користувачем пароль із збереженим у базі. Для цього виконується повторне хешування введеного пароля з використанням збереженої солі, і результат порівнюється з оригінальним хешем.

```
public bool VerifyPassword(string password, string hash, string salt)
{
    var convertedSalt = Convert.FromBase64String(salt);
    var hashToCompare = Rfc2898DeriveBytes.Pbkdf2(password, convertedSalt, Iterations,
        _hashAlgorithm, KeySize);

    return CryptographicOperations.FixedTimeEquals(hashToCompare,
        Convert.FromHexString(hash));
}
```

Другим є TokenService, який відповідає за створення та валідацію JWT-токенів. Метод GenerateAccessToken створює JWT-токен. Для генерації використовується HMAC SHA256 як алгоритм підпису, а термін дії токена встановлюється на 180 хвилин. Окремо реалізовано метод GenerateRefreshToken, який створює довгий випадковий рядок, що використовується для оновлення токена доступу без повторної авторизації.

```
public string GenerateAccessToken(User user)
{

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

```

var claims = new List<Claim>
{
    new (ClaimTypes.Email, user.Email)
};
return GenerateToken(claims);
}

public string GenerateRefreshToken()
{
    var randomNumber = new byte[64];
    using var rng = RandomNumberGenerator.Create();
    rng.GetBytes(randomNumber);

    return Convert.ToBase64String(randomNumber);
}

```

Для перевірки валідності токена використовується метод `GetPrincipalClaims`.

```

public ClaimsPrincipal GetPrincipalClaims(string token)
{
    var tokenValidationParameters = new TokenValidationParameters
    {
        ValidateAudience = true,
        ValidateIssuer = true,
        ValidateIssuerSigningKey = true,
        ValidAudience = _audience,
        ValidIssuer = _issuer,
        RequireSignedTokens = false,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_key)),
        RequireExpirationTime = false,
        ValidateLifetime = false
    };

    var tokenHandler = new JwtSecurityTokenHandler();
    var principal = tokenHandler.ValidateToken(token, tokenValidationParameters, out var
securityToken);

    var jwtSecurityToken = (JwtSecurityToken)securityToken;
    bool isAlgMatch = jwtSecurityToken.Header.Alg.Equals(
        SecurityAlgorithms.HmacSha256,
        StringComparison.InvariantCultureIgnoreCase
    );

    if (jwtSecurityToken == null || !isAlgMatch)
    {
        throw new SecurityTokenException("Invalid token");
    }

    return principal;
}

```

Розглянемо реалізацію логіки авторизації користувача у вигляді команди `LoginUesrCommand` (Додаток В), яка містить два параметри – електронну пошту та пароль, які передає користувач під час входу. Відповідальність за обробку цієї команди покладено на клас `LoginUesrCommandHandler`

```

public async Task<AuthenticatedUserDTO> Handle(LoginUserCommand request,
Cancellation token cancellationToken)
{
    var userByEmail = await _userRepository.GetUserByEmailAsync(request.Email);
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

```

if (userByEmail is null)
    throw new Exception("User with provided email does not exist");

var verifyResult =
    _passwordService.VerifyPassword(request.Password, userByEmail.HashedPassword,
userByEmail.Salt);
if (!verifyResult)
    throw new Exception("Login failed");

var accessToken = _tokenService.GenerateAccessToken(userByEmail);

return new AuthenticatedUserDTO
{
    AccessToken = accessToken,
    FirstName = userByEmail.FirstName,
    LastName = userByEmail.LastName,
    RoleName = userByEmail.Role!.Name
};
}

```

Висновки до розділу 2

У другому розділі було здійснено детальне проєктування веб-орієнтованої системи для відстеження технічного обслуговування автомобілів на СТО. Поетапно розглянуто ключові елементи архітектури програмного забезпечення: варіанти використання системи, об'єктно-орієнтовану модель, принципи роботи, структуру бази даних та технічну реалізацію клієнтської і серверної частин.

У процесі розробки було проаналізовано ролі користувачів та їхню взаємодію із системою, що дало змогу сформулювати чітке уявлення про функціональні можливості кожного типу користувача на основі побудованих UML-діаграм. Використання об'єктно-орієнтованого підходу разом із побудовою діаграми сутностей сприяло впорядкуванню логіки роботи додатку та полегшило проєктування архітектури. Структура бази даних була сформована на основі чітко визначених зв'язків між сутностями, що забезпечує цілісність даних, логічну узгодженість та можливість масштабування системи в майбутньому.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

3.1. Порядок встановлення та налаштування параметрів системи

Для того щоб повноцінно запустити веб-орієнтовану систему для відстеження технічного обслуговування автомобілів, спочатку необхідно правильно встановити та налаштувати як серверну, так і клієнтську частину додатку. Система побудована за сучасною архітектурою SPA (Single Page Application), де фронтенд та бекенд є повністю розділеними, що дає змогу окремо розгортати обидві частини та масштабувати їх у разі потреби. Бекенд написаний на мові програмування C# з використанням .NET і реалізований згідно з принципами Clean Architecture. Такий підхід дозволяє чітко відділити бізнес-логіку від інфраструктурної реалізації та зробити систему більш підтримуваною і масштабованою.

У серверній частині розроблено чотири основні проєкти: перший — це GraduationWork.Application, в якому зосереджена вся бізнес-логіка, сервіси, контракти та шаблони для команд і запитів у стилі CQRS; другий — GraduationWork.Domain, де зберігаються всі сутності, що описують бізнес-предметну область, включно з переліками (enum) та іншими логічними об'єктами; третій — GraduationWork.Infrastructure, який відповідає за доступ до бази даних, реалізацію зовнішніх сервісів, таких як обробка паролів, робота з токенами, а також підключення до платіжної системи LiqPay; і четвертий — GraduationWork.WebApi, це безпосередньо веб-сервер, який обробляє HTTP-запити, надає REST API та відповідає за авторизацію і взаємодію з клієнтською частиною.

Для запуску серверної частини насамперед потрібно встановити середовище розробки .NET SDK (версія 8.0 або сумісна) і переконатися, що на комп'ютері встановлений Microsoft SQL Server, адже база даних системи реалізована через Entity Framework Core, який працює з MS SQL. Після цього відкривається рішення в середовищі Visual Studio або будь-якому іншому зручному редакторі, де потріб-

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

но вказати у файлі `appsettings.json` правильний рядок з'єднання до локального SQL Server. Наступним кроком є виконання команди створення бази даних, яка застосує всі необхідні міграції. Це можна зробити за допомогою команди через NuGet Package manager console `update-database`, де вказується проєкт з інфраструктурною логікою. Після успішного оновлення бази даних можна запускати веб-сервер, який працює на портах 5000 (HTTP) або 5001 (HTTPS) за замовчуванням. Додаток реалізує автентифікацію через JWT токени, тому для користувача необхідно пройти авторизацію, щоб отримати токен доступу до захищених маршрутів.

Фронтенд частина додатку розроблена з використанням React – сучасної бібліотеки для побудови інтерфейсів, а також Ant Design – системи компонентів, що значно пришвидшує розробку UI. Проєкт збирається за допомогою Vite, що забезпечує дуже швидку розробку та збірку. Для запуску клієнтської частини необхідно встановити Node.js (рекомендована версія — 18 або новіша). Далі в директорії клієнта виконується команда `npm install`, яка завантажує всі необхідні залежності. У файлі `.env` потрібно вказати адресу веб-серверу бекенду, наприклад `VITE_API_URL=https://localhost:5001/api`. Після цього команда `npm run dev` запускає фронтенд у режимі розробки.

При переході на цей сайт відкривається інтерфейс користувача, який працює з API, надсилає запити з JWT токенами в заголовках та виконує всі необхідні дії, пов'язані із замовленнями, клієнтами, менеджерами та іншими модулями системи.

Важливим компонентом системи є інтеграція з платіжною системою LiqPay. Для цього реалізовано окремий сервіс, який працює на інфраструктурному рівні. Ключі доступу `public_key` та `private_key`, які передаються у конфігураційних файлах або встановлюються як змінні середовища. Під час оплати створюється спеціальне зашифроване посилання на сторінку LiqPay, після завершення якої інформація про платіж зберігається у базу даних. Це дозволяє контролювати процеси оплати послуг або товарів у системі та спрощує фінансову частину для кінцевих користувачів.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Таким чином, вся система легко розгортається локально, з чітким розділенням на серверну та клієнтську частини, з підтримкою авторизації, бази даних, а також інтеграцією із зовнішньою платіжною платформою. Такий підхід дозволяє використовувати її як у навчальних цілях, так і як прототип або основу для впровадження в реальну роботу СТО.

3.2. Структура інтерфейсу веб-орієнтованої системи для відстеження технічного обслуговування автомобілів

Інтерфейс користувача вебзастосунку для відстеження технічного обслуговування автомобілів є ключовим елементом забезпечення ефективної та інтуїтивно зрозумілої взаємодії між користувачами та функціоналом системи. Серед основних груп користувачів — власник, адміністратори, менеджери та механіки, кожна з яких має доступ до окремих інтерфейсних компонентів відповідно до своєї ролі. Зручність навігації, зрозумілість розміщення елементів та швидкий доступ до головних функцій, таких як створення замовлень, перегляд стану обслуговування, додавання запчастин або реєстрація клієнтів, напрямку впливають на продуктивність роботи з системою.

Інтерфейс реалізовано відповідно до принципів сучасного UX/UI-дизайну з використанням бібліотеки Ant Design, що дозволяє дотримуватись єдиного стилю для всіх візуальних компонентів. У цьому підрозділі розглянуто структуру основних інтерфейсів вебзастосунку, описано порядок роботи з ними та пояснено, яким чином реалізовано користувацьку логіку взаємодії з системою технічного обслуговування.

Для початку, щоб отримати доступ до ресурсів додатку потрібно увійти в систему (рис.3.1.)

Рисунок 3.1. Форма логіну у систему

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішної аутентифікації користувач потрапляє на головну сторінку (рис.3.2.) де він може побачити календар для планування замовлень. У календарі можливе відображення усіх ресурсів відразу та можливість вибрати один (рис. 3.2.). Також можливий вибір перегляду календаря по тижню (рис.3.2.) та по дню (рис.3.4.). Кожен день у календарі розділений на робочі години. Для створення та планування замовлення на конкретний день та час потрібно вибрати проміжок часу. Тоді з'являється форма для створення замовлення (рис.3.5.) де можна призначити менеджера та механіка.

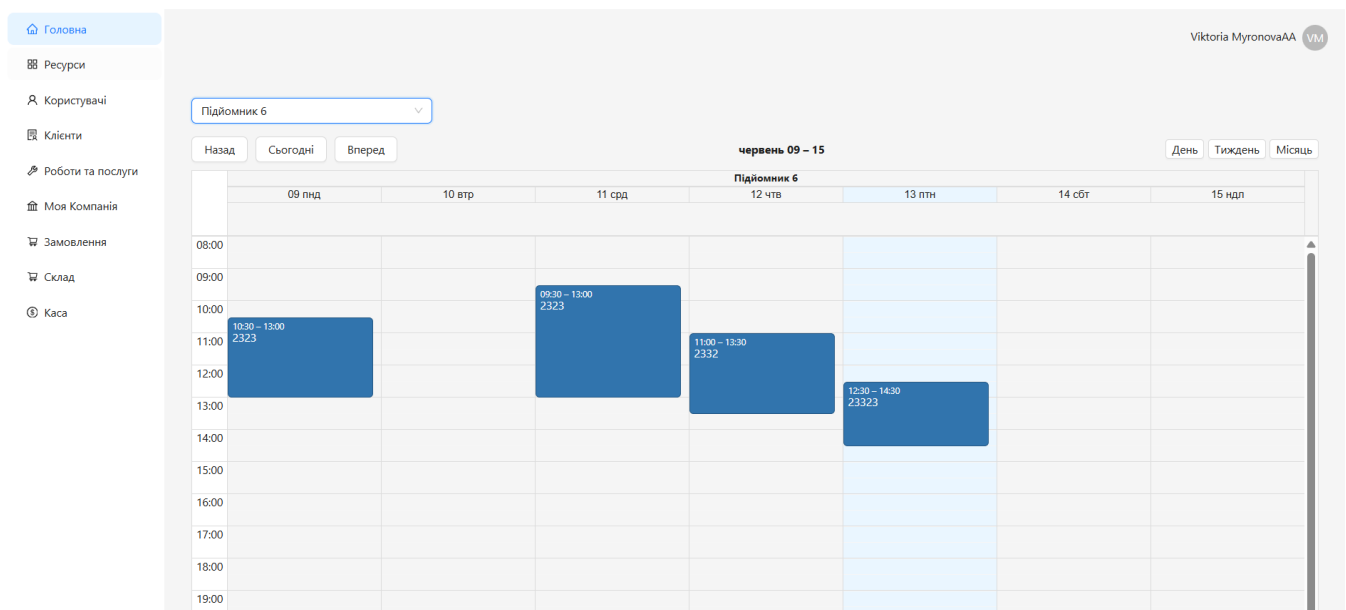


Рисунок 3.2. Головна сторінка додатку

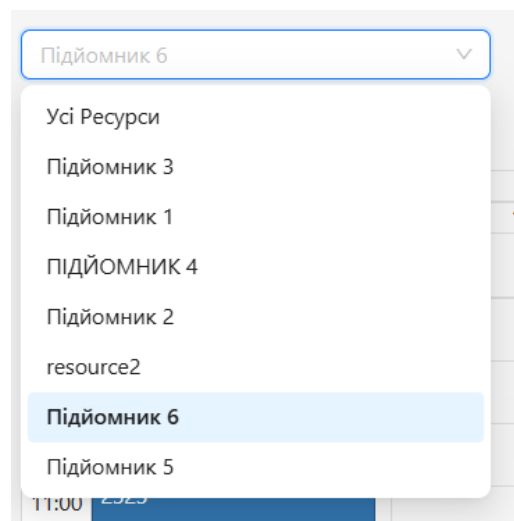


Рисунок 3.3. Вибір ресурсів у календарі

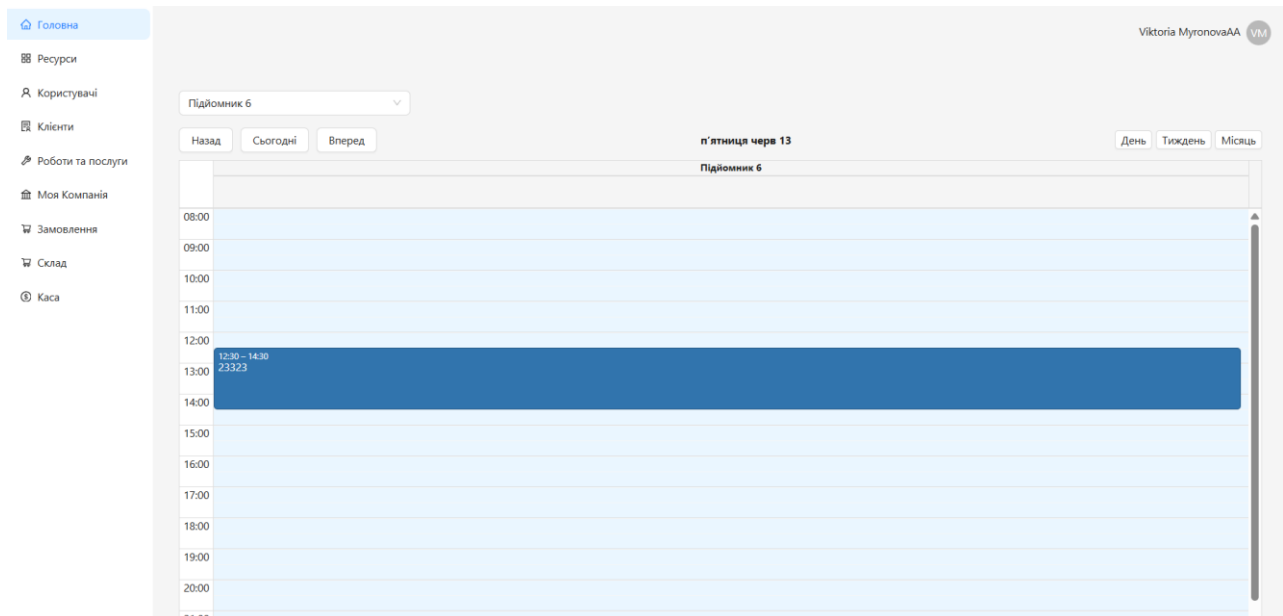


Рисунок 3.4. Відображення календаря по дню

Створити Замовлення

* Vin

Введіть Vin

* Марка

Введіть Марка

* Модель

Введіть Модель

* Рік

Введіть Рік

* Нотатки

Введіть Нотатки

* Дата Початку

Введіть Дата Початку

* Дата Завершення

Введіть Дата Завершення

* Статус

Оберіть статус

* Менеджер

Оберіть менеджера

* Клієнт

Оберіть клієнта

* Механік

Оберіть механіка

* Ресурс

Оберіть ресурс

Відмінити

Створити

Рисунок 3.5. Форма створення замовлення

Також є можливість виходу з системи. Для цього потрібно натиснути на ім'я користувача у верхньому правому кутку і натиснути вийти (рис.3.6.).

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

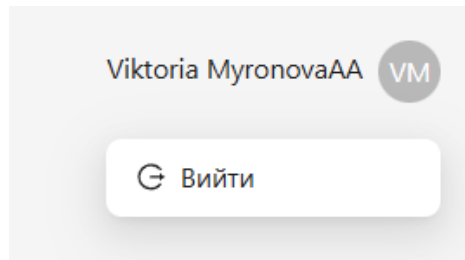


Рисунок 3.6. Вихід з системи

Далі ми можемо перейти до сторінки ресурсів. Там ми можемо побачити відображення всіх ресурсів які доступні у системі у вигляді таблиці (рис.3.7.). Є можливість створити новий ресурс через кнопку «Створити Ресурс». Також у колонці «Дії» доступні редагування та видалення ресурсів (рис.3.8.).

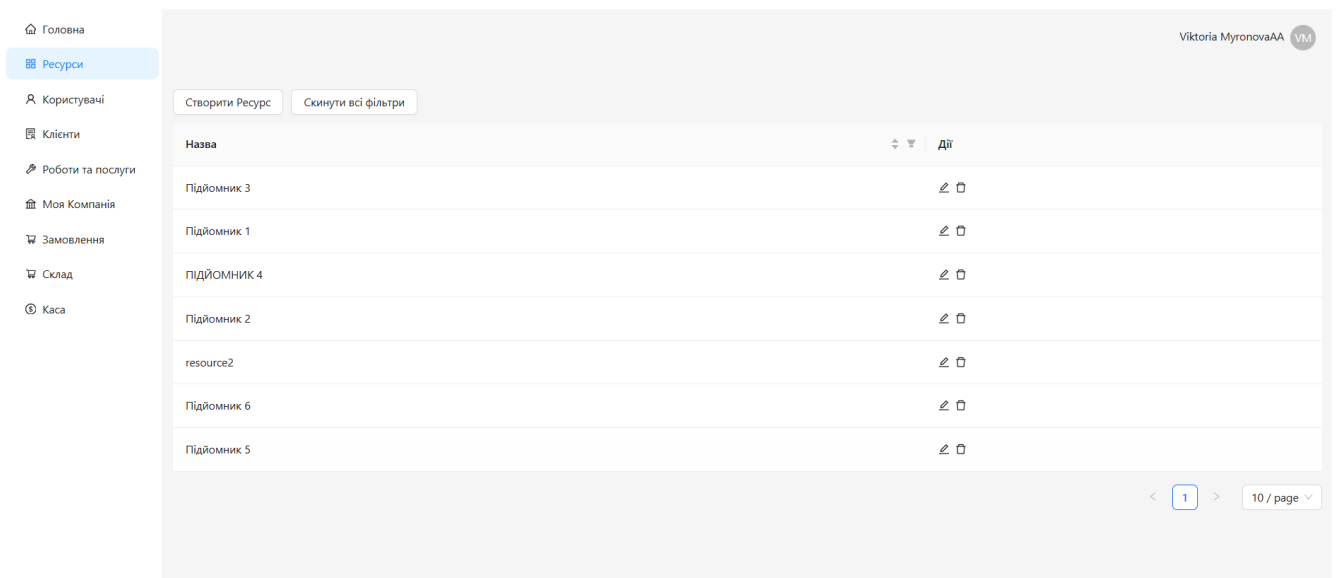


Рисунок 3.7. Сторінка Ресурси

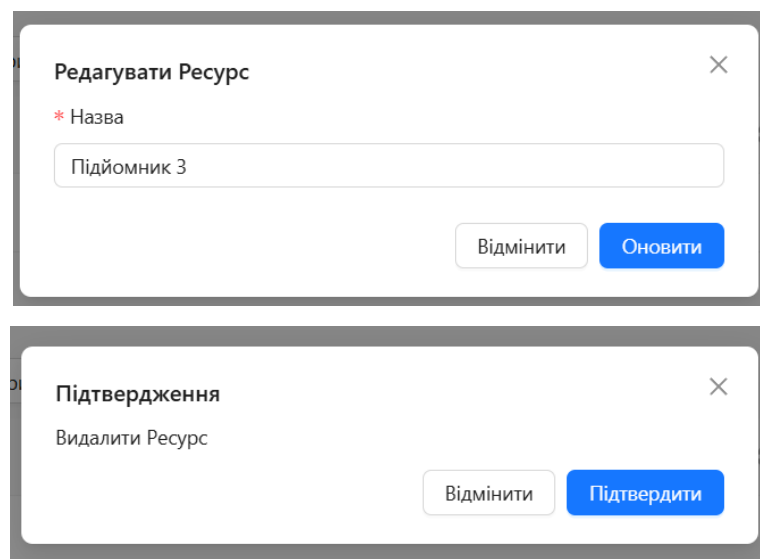


Рисунок. 3.8. Форма редагування та видалення ресурсу

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Також у додатку є можливість сортування та фільтрації (рис.3.9-3.10.).

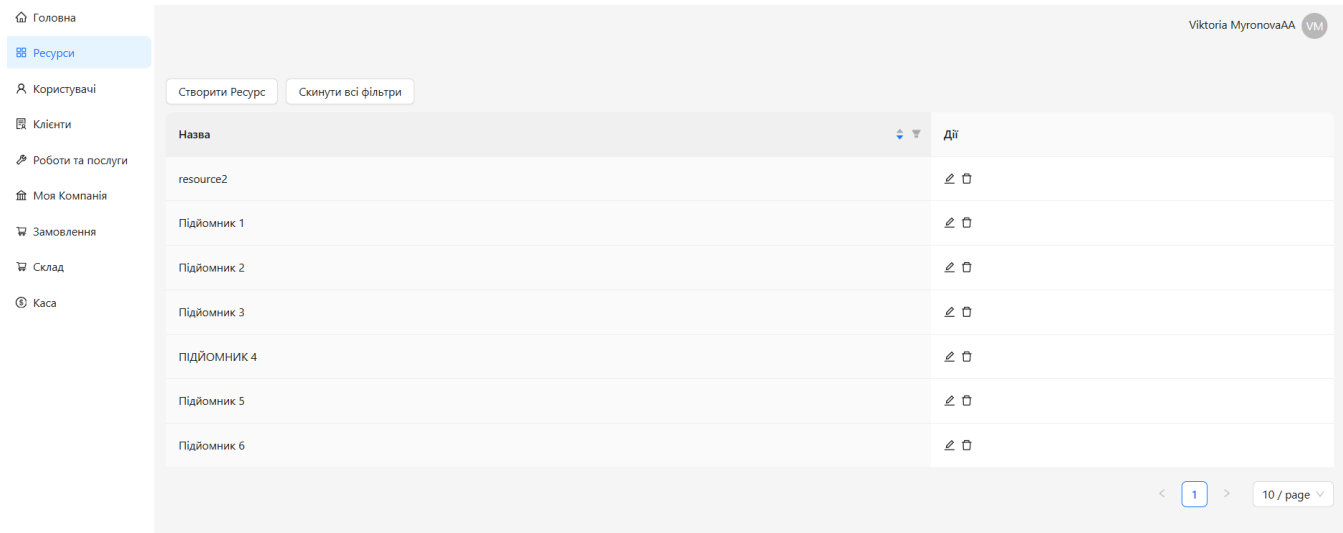


Рисунок 3.9. Застосоване сортування по назві Ресурсів

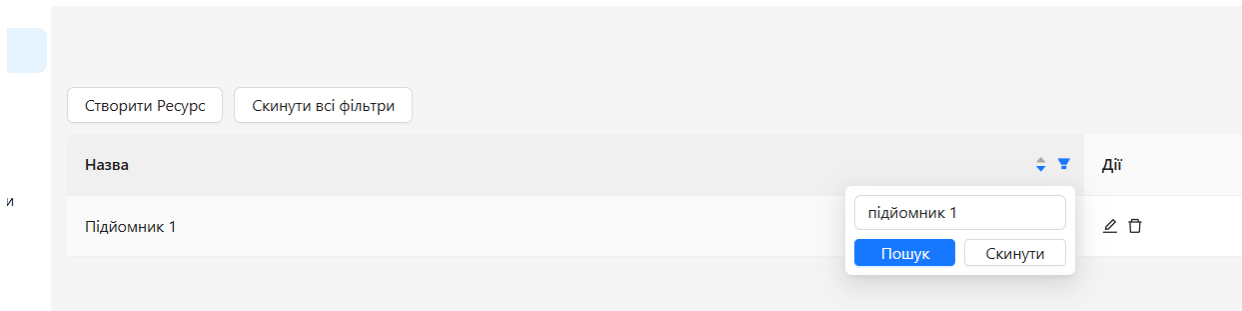


Рисунок 3.10. Застосована фільтрація по назві Ресурсів

Далі ми можемо перейти на сторінку Користувачі і побачити таблицю користувачів системи з ролями (рис.3.11.). У таблиці є можливість сортування та фільтрації. У колонці Дії є можливість видалити та редагувати користувача а також створити користувача та скинути всі застосовані фільтри.

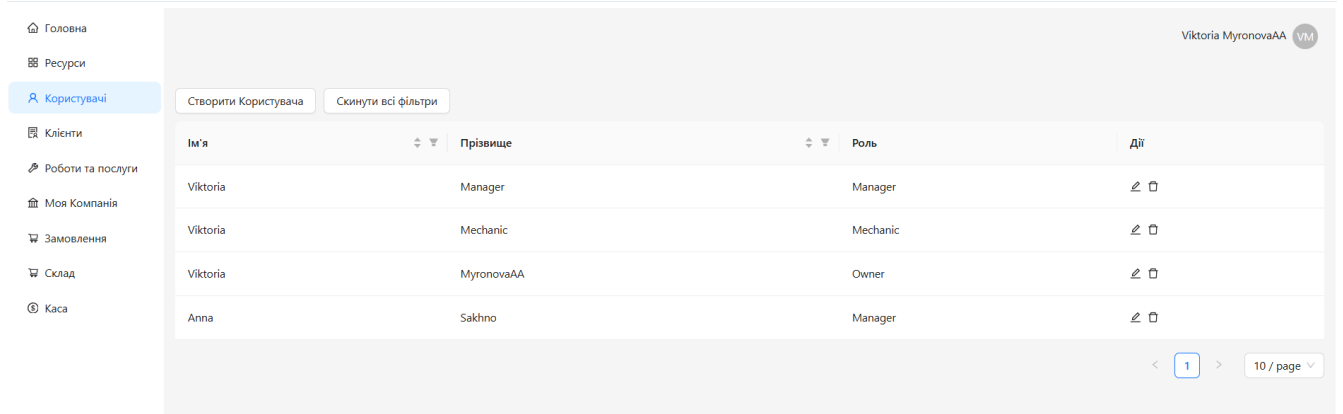


Рисунок 3.11. Сторінка користувачів системи

Переходимо до сторінки Клієнти. Тут відображена таблиця всіх клієнтів з можливістю створення, редагування видалення, фільтрації та сортування (рис.3.12.).

Головна

Ресурси

Користувачі

Клієнти

Роботи та послуги

Моя Компанія

Замовлення

Склад

Каса

Створити Клієнта

Скинути всі фільтри

Ім'я	Прізвище	Email	Адреса	Дії
Qwerty	Qwerty	qwerty@gmail.com	вул Житомирська, 81	
Vika	Myronova	ipz212_mvl_cust@student.ztu.edu.ua	Zhytomyr	

<1>

10 / page

Рисунок 3.12. Сторінка Клієнти

Далі сторінка Роботи та послуги. Бачимо таблицю всіх послуг з можливістю створення, редагування видалення, фільтрації та сортування (рис.3.13.)

Головна

Ресурси

Користувачі

Клієнти

Роботи та послуги

Моя Компанія

Замовлення

Склад

Каса

Viktoria MyronovaAA

VM

Створити Виконану роботу

Скинути всі фільтри

Назва роботи	Ціна	Дії
Полірування передніх фар	850	✎ ✖
Антидоць	650	✎ ✖
Встановлення LED ламп у габарити	800	✎ ✖
Полірування задніх ліхтарів	700	✎ ✖
Встановлення протитуманних фар	2500	✎ ✖

< 1 >

10 / page

Рисунок 3.13. Сторінка Роботи та послуги

Далі сторінка Моя компанія. Тут ми можемо заповнити інформацію про нашу компанію, до прикладу назву компанії, опис, адреса, місто розташування, номер тееофону, email компанії та вказати Прізвище Ім'я По батькові директора компанії (рис.3.14.).

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Інформація про компанію

* Назва

Товариство з обмеженою відповідальністю "AutoRepair"

Опис

Товариство з обмеженою відповідальністю "AutoRepair"

* Адреса

вул. Цетайрівська, 6

* Місто

Київ

Телефон

+380000000000

Email

autorepair@gmail.com

* Директор

Стефанчук Микола Петрович

Зберегти

Рисунок 3.14. Сторінка Моя компанія

Переходимо до головного модуля нашого додатку – Замовлення. На цій сторінці ми бачимо всі створені та заплановані замовлення (рис.3.15.). Є можливість створення замовлення через кнопку «Створити замовлення», сортування, фільтрація по колонках. Окремо можна виділити у цій таблиці це колонку Дії. Адміністратор системи має доступ до всіх відображених дій. Перша дія це завантаження документа «Акт виконаних робіт», який генерується по шаблону після натискання на кнопку завантаження (рис 3.16.)

Створити Замовлення

Скинути всі фільтри

Номер замовлення	Дата початку	Дата закінчення	Механік	Менеджер	Користувач	Ресурси	Статус	Дії
TIQUDRJYKK	09.06.2025, 09:00:00	09.06.2025, 10:30:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 1	Нове	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
M8ZRPAP33K	11.06.2025, 09:30:00	11.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	Діагностика	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
SRHTPQD3KK	09.06.2025, 10:30:00	09.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	В роботі	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
WKHCZAJAVE	12.06.2025, 09:30:00	12.06.2025, 12:00:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 2	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
W3N4OIOZKU	13.06.2025, 12:30:00	13.06.2025, 14:30:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
KAY2NGDORU	11.06.2025, 10:30:00	11.06.2025, 13:30:00	Viktoria Mechanic	Anna Sakhno	Qwerty Qwerty	ПІДЙОМНИК 4	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
HOIWSKWWAQ	12.06.2025, 11:00:00	12.06.2025, 13:30:00	Viktoria Mechanic	Anna Sakhno	Qwerty Qwerty	Підйомник 6	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
HN0IMQ6VTU	11.06.2025, 00:00:00	11.06.2025, 05:00:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 1	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇
WFOVOMJIUP	12.06.2025, 11:00:00	12.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 1	Закрито	⬇ ⬆ ⬇ ⬇ ⬇ ⬇

1

10 / page

Рисунок 3.15. Сторінка Замовлення

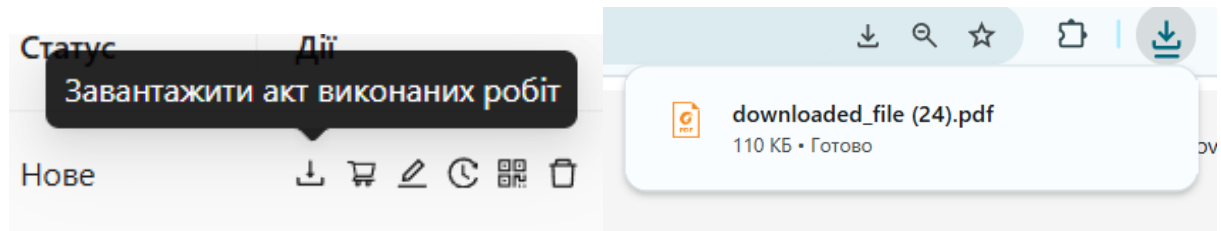


Рисунок 3.16. Завантаження документа «Акт виконаних робіт»

Після завантаження ми може переглянути документ та які роботи і продукти були використані у даному замовленні а також підрахунок загальної вартості по цим роботам та продуктах (рис.3.17.).

1 / 1 | - 100% + | [Icon] [Icon]

ЗАТВЕРДЖУЮ

Виконавець

Товариство з обмеженою відповідальністю "AutoRepair"

Стефанчук Микола Петрович

ЗАТВЕРДЖУЮ

Замовник

Qwerty Qwerty

qwerty@gmail.com

13.06.2025

м.Київ

АКТ наданих послуг

Ми, що нижче підписалися, представник Замовника з одного боку, і представник Виконавця з іншого боку, склали цей акт про те, що за договором від 09.06.2025 р.:

Виконавцем були надані такі послуги:

Вид послуг	Кількість	Ціна без ПДВ	Сума без ПДВ
Встановлення LED ламп у габарити	1	800,00	800,00

Матеріал	Кількість	Ціна без ПДВ	Сума без ПДВ
Тосол	1	200,00	200,00

Разом: 1000,00

Сума ПДВ: 200,00

Разом із ПДВ: 1200,00

Замовник не має претензій до Виконання стосовно наданих послуг.

Від Виконавця

Від Замовника

Директор

Замовник

Стефанчук Микола Петрович

Qwerty Qwerty

Товариство з обмеженою відповідальністю "AutoRepair"

вул Житомирська, 81

м. Київ, вул. Петгайрівська, 6

qwerty@gmail.com

Рисунок 3.17. Згенерований документ «Акт виконаних робіт»

Наступна дія в межах нашого замовлення – це керування елементами (рис.3.18.). Тобто це додавання виконаних робіт та продуктів зі складу до поточного замовлення. Присутня перевірка чи є достатня кількість продуктів на складі.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмостра О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Керувати елементами

Виконані Роботи

Кількість	Назва	Ціна	Дії
1	Встановлення LED ламп у габарити	800.00	<a>Edit

Продукти

Недостатня кількість продукту 'Тосол'. Наявна кількість: 1.

Кількість	Продукт	Ціна	Дії
3	Тосол	600.00	<a>Редагувати

Загальна вартість виконаних робіт: 800.00

Загальна вартість продуктів: 600.00

Загальна вартість: 1400.00

Ціна з податками: 1680.00

+ Додати виконану роботу

+ Додати Продукт

Зберегти

Закрити

Рис. 3.18. Додавання виконаних робіт та продуктів до замовлення

Наступна дія – це редагування замовлення (рис. 3.19.). Тут ми можемо змінювати статус замовлення, менеджера, механіка із подальшим записом у історію замовлення.

* Нотатки

.....

* Дата Початку

2025-06-09 09:00

* Дата Завершення

2025-06-09 10:30

* Статус

Нове

* Менеджер

Viktoria Manager

* Клієнт

Vika Myronova

* Механік

Viktoria Mechanic

* Ресурс

Підйомник 1

Відмінити

Оновити

Рисунок 3.19. Редагування замовлення

Наступна кнопка у колонці Дії відповідає за відображення історії замовлення (рис.3.20.). Сюди записується історія змін до поточного замовлення до прикладу Замовлення створено, які продукти чи виконані роботи були додані, про зміну Менеджера та Механіка, про оплату замовлення. Редагувати історію замовлення немає можливості.

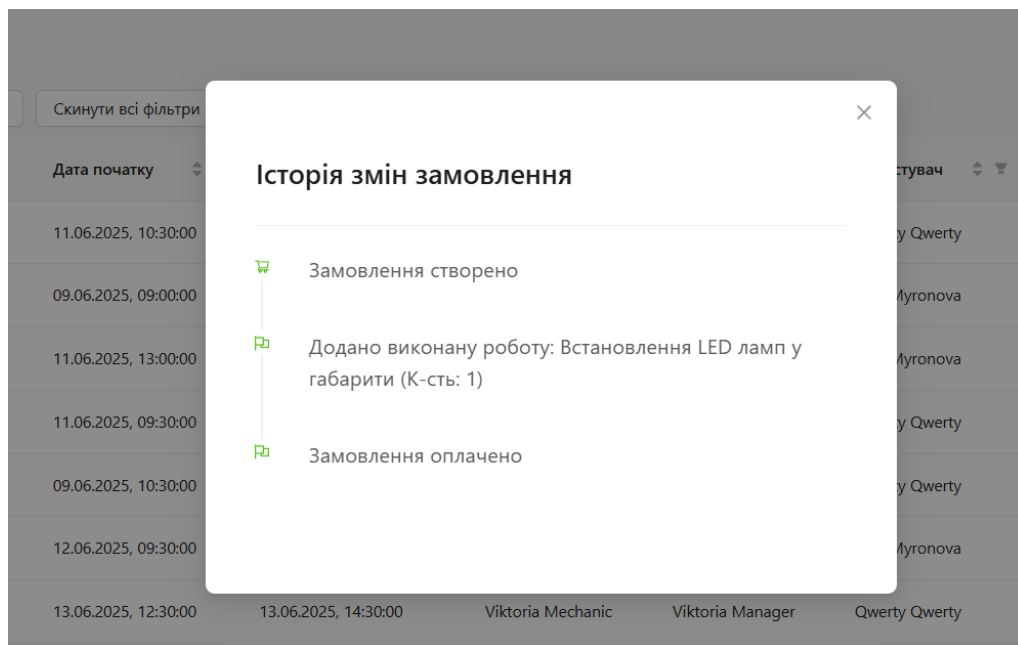


Рисунок 3.20. Відображення історії замовлення

Далі ми можемо перейти до генерації QR-коду для оплати замовлення. При натисканні на відповідну іконку відкривається модальне вікно із згенерованим QR-кодом та можливістю перейти до оплати через посилання (рис.3.21.).

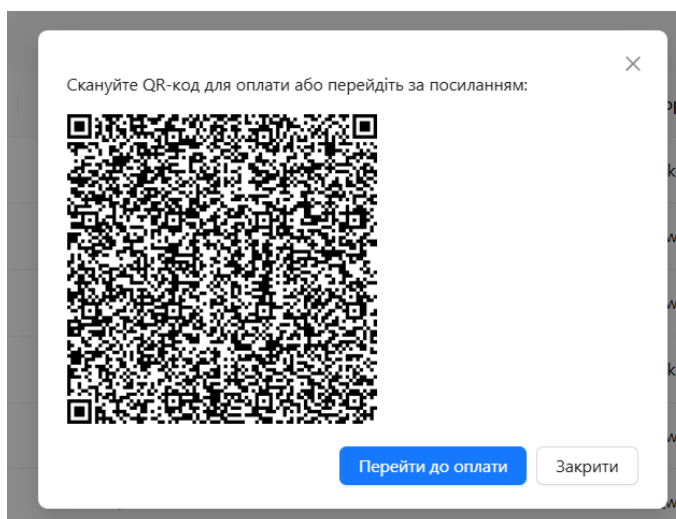


Рисунок 3.21. QR-код для оплати замовлення

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

При натисканні на кнопку «Перейти до оплати» відкривається посилання у новому вікні де ми можемо оплатити замовлення зручним способом (рис.3.22.).

QR-код для оплати

Використовуйте Privat24

Тестовий режим

LIQPAY

Дані про оплату

Оплата замовлення №be53c8a5-a8d5-4d0e-98c7-b6b8638325cc

До сплати: **780.00 UAH**

24 Pay

Google Pay

або

Картка

Приват24

Рахунок

Номер картки

0000 0000 0000 0000

Термін дії

MM/YY

CVV2

☐ Відправити квитанцію на e-mail

Рисунок 3.22. Сторінка оплати замовлення

Після успішної оплати з'являється повідомлення та можливість повернутися до нашого сайту (рис.3.23.).

Тестовий режим

✓

Дякуємо!

Платіж успішно відправлено.

ID: 2660502043

Дата: 13.06.2025 03:44:01

Сума: **300000.00 UAH**

E-mail для отримання квитанції

Введіть e-mail

Надіслати квитанцію

Завантажити квитанцію

Повернутися до сайту

Рисунок 3.23. Повідомлення про успішну оплату

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Наступне, що ми можемо зробити у межах одного замовлення це видалити замовлення (рис.3.24).

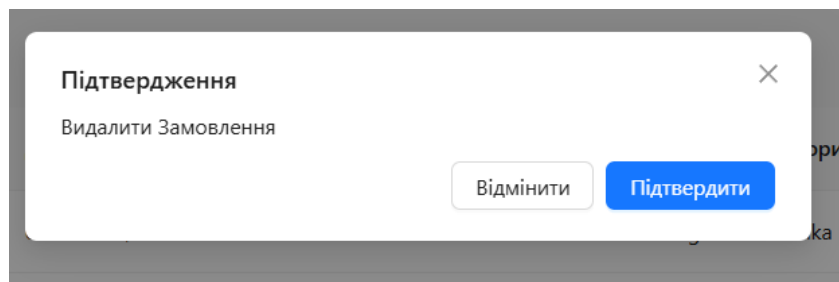


Рисунок 3.24. Підтвердження видалення замовлення

Далі переходимо до сторінки Склад. Тут ми можемо побачити які продукти є у нас на складі, створити новий продукт, відсортувати та відфільтрувати продукти, редагувати та видалити (рис.3.25-3.27).

Головна	Viktoria MyronovaAA VM							
Ресурси	Створити Продукт Скинути всі фільтри							
Користувачі	Назва	Кількість	Ціна закупівлі	Ціна продажу	Тип	Дії		
Клієнти	Амортизатор підвіски 19029443 BILSTEIN	50	1800	2085	Запчастина			
Роботи та послуги	Тосол	1	100	200	Запчастина			
Моя Компанія	Олива моторна hyundai/kia super extra gasoline 5w-30, 4л.	10	1860	2099	Запчастина			
Замовлення	Фільтр	2	100	200	Запчастина			
Склад	Ремінь полікліновий	3	1000	1182	Запчастина			
Каса	Підставка карбюратора	5	600	845	Запчастина			
	Підшипник колісний	10	400	529	Запчастина			
	Фільтр масляний 1001150004 MEYLE	15	150	250	Запчастина			
	< 1 > 10 / page							

Рисунок 3.25. Сторінка Складу

Створити Продукт

Назва

Амортизатор підвіски 19029443 BILSTEIN

Кількість

50

Ціна закупівлі

1800

Ціна продажу

2085

Тип

Запчастина

Відмінити

Створити

Рисунок 3.26. Форма створення продукту

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Редагувати Продукт

* Назва

Підшипник колісний

* Кількість

10

* Ціна закупівлі

400

* Ціна продажу

529

* Тип

Запчастина

Відмінити

Оновити

Рисунок 3.27. Форма редагування продукту

Останній модуль у нашій системі це Каса. На цій сторінці ми бачимо назву каси та її баланс а також таблицю з транзакціями у нашій системі (рис. 3.28). Тут ми можемо створити транзакцію на приклад для оплати замовлення. Також при додаванні продуктів та оплати замовлення створюються транзакції та відображаються тут. Можливі також сортування, фільтрація та редагування транзакцій.

Головна

Ресурси

Користувачі

Клієнти

Роботи та послуги

Моя Компанія

Замовлення

Склад

Каса

Viktoria MyronovaAA VM

Каса №1

Поточний баланс: 50120 ₴

Створити Транзакцію

Скинути всі фільтри

Дата та час	Сума	Тип	Метод оплати	Опис	Дії
2025-06-13T00:04:39.0406924	4000	Expense	Cash	Додано продукт: Підшипник колісний	✎ ✕
2025-06-13T00:03:31.0670097	3000	Expense	Cash	Додано продукт: Підставка карбюратора	✎ ✕
2025-06-13T00:02:27.6902159	3000	Expense	Cash	Додано продукт: Ремінь полікліновий	✎ ✕
2025-06-13T00:01:04.6438508	18600	Expense	Cash	Додано продукт: Олива моторна hyundai/kia super extra gasoline 5w-30, 4л.	✎ ✕
2025-06-13T23:59:06.0639391	90000	Expense	Cash	Додано продукт: Амортизатор підвіски 19029443 BILSTEIN	✎ ✕
2025-06-12T23:56:29.9677338	2250	Expense	Cash	Додано продукт: Фільтр масляний 1001150004 MEYLE	✎ ✕
2025-06-11T21:33:05.7322798	780	Income	Card	Оплата замовлення WKHCZAJAVE	✎ ✕

Рисунок 3.28.Сторінка Каса

У нашому додатку реалізовані ролі доступу. В залежності від ролі визначається що буде доступно користувачу а що ні. До прикладу роль Механік. Йому доступні тільки такі модулі: Головна, Клієнти, Роботи та послуги, Замовлення, де відображаються тільки ті замовлення, які призначені механіку (рис.3.30), Склад та Каса (рис.3.29). У механіка немає доступу до зміни чи видалення, але є доступ до зміни статусу замовлення.

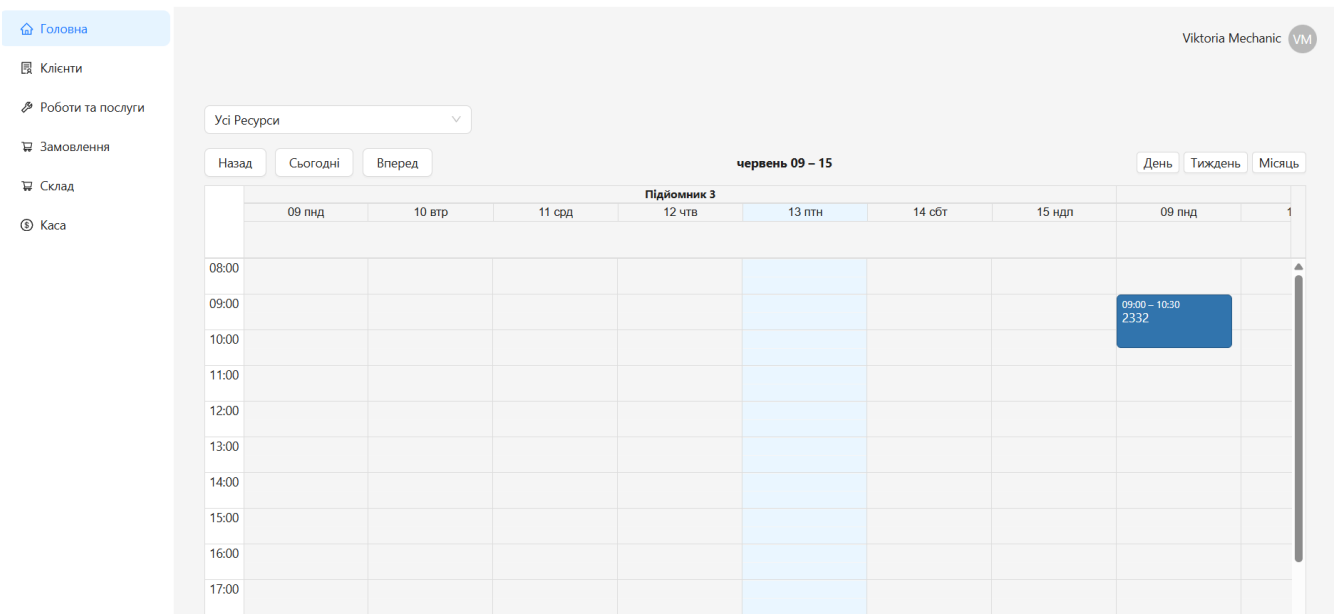


Рисунок 3.29.Сторінка Головна для ролі Механік

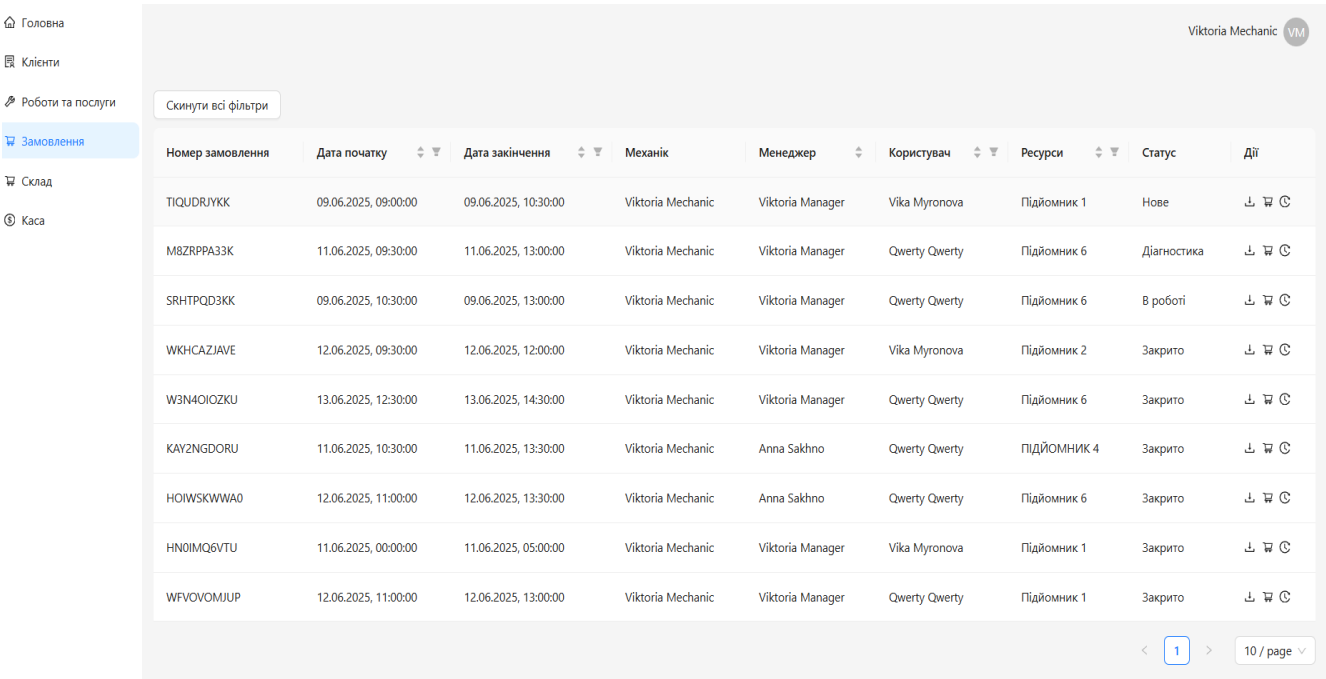


Рисунок 3.30. Відображення замовлень призначених ролі Механік

Для ролі Менеджер доступні сторінки такі як і у ролі Механік але ще плюс сторінка Ресурсів. Менеджер бачить тільки призначені йому замовлення (рис 3.31). Має можливість створювати, редагувати замовлення, клієнтів, повний CRUD для послуг і ресурсів, перегляд транзакцій (рис 3.32). Також менеджер не має змоги створювати QR-код для оплати замовлення.

Головна	Viktoria Manager VM
Ресурси	Створити ЗамовленняСкинути всі фільтри
Клієнти	
Роботи та послуги	
Замовлення	
Склад	
Каса	

Номер замовлення	Дата початку	Дата закінчення	Механік	Менеджер	Користувач	Ресурси	Статус	Дії
TIQUDRJYKK	09.06.2025, 09:00:00	09.06.2025, 10:30:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 1	Нове	Дії
M8ZRPPA33K	11.06.2025, 09:30:00	11.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	Діагностика	Дії
SRHTPQD3KK	09.06.2025, 10:30:00	09.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	В роботі	Дії
WKHCZAJAVE	12.06.2025, 09:30:00	12.06.2025, 12:00:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 2	Закрито	Дії
W3N4OIOZKU	13.06.2025, 12:30:00	13.06.2025, 14:30:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 6	Закрито	Дії
HN0IMQ6VTU	11.06.2025, 00:00:00	11.06.2025, 05:00:00	Viktoria Mechanic	Viktoria Manager	Vika Myronova	Підйомник 1	Закрито	Дії
WFVOVOMJUP	12.06.2025, 11:00:00	12.06.2025, 13:00:00	Viktoria Mechanic	Viktoria Manager	Qwerty Qwerty	Підйомник 1	Закрито	Дії

Рисунок 3.31. Доступні сторінки для ролі менеджера; відображення тільки своїх замовлень

Головна	Viktoria Manager VM
Ресурси	
Клієнти	
Роботи та послуги	
Замовлення	
Склад	
Каса	

Каса №1

Поточний баланс: 50012 ₴

Дата та час	Сума	Тип	Метод оплати	Опис	Дії
2025-06-13T00:04:39.0406924	4000	Exponce	Cash	Додано продукт: Підшипник колісний	
2025-06-13T00:03:31.0670097	3000	Exponce	Cash	Додано продукт: Підставка карбюратора	
2025-06-13T00:02:27.6902159	3000	Exponce	Cash	Додано продукт: Ремінь полікліновий	
2025-06-13T00:01:04.6438508	18600	Exponce	Cash	Додано продукт: Олива моторна hyundai/kia super extra gasoline 5w-30, 4л.	
2025-06-12T23:59:06.0639391	90000	Exponce	Cash	Додано продукт: Амортизатор підвіски 19029443 BILSTEIN	
2025-06-12T23:56:29.9677338	2250	Exponce	Cash	Додано продукт: Фільтр масляний 1001150004 MEYLE	

Рисунок 3.32. Дії для менеджера на сторінці транзакцій недоступні

Для клієнта реалізована можливість дізнатися інформацію про замовлення за номером замовлення без попередньої реєстрації та авторизації (рис.3.33).

Інформація про замовлення

Ідентифікатор замовлення

7T866UJXN0

Отримати деталі замовлення

Деталі замовлення

Статус: Закрито

Ім'я клієнта: Qwerty Qwerty

Ім'я менеджера: Viktoria Manager

Ціна з податками: 300000 грн.

Ціна без податків: 250000 грн.

Оплатити

Рисунок 3.33. Сторінка інформації про замовлення для клієнта

3.3. Тестування роботи програмного комплексу

Для забезпечення стабільності, коректності та надійної роботи розробленого програмного продукту було проведено функціональне тестування основних модулів системи. Тестування виконувалося вручну згідно з типовими сценаріями використання, які охоплюють усі ключові функціональні можливості застосунку. Для кожного сценарію були визначені передумови, послідовність дій користувача, очікуваний результат та фактичний результат після виконання дій. Усі перевірки проводилися на останній стабільній збірці застосунку в умовах, наближених до реального середовища експлуатації.

1. Авторизація користувача у системі

Користувач успішно авторизувався в системі після введення валідних облікових даних, зокрема email та пароля, на сторінці логіну. Після натискання кнопки «Увійти» система виконала перенаправлення на головну сторінку додатку з календарем, що відповідає очікуваному результату.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

2. Планування замовлення в календарі

У процесі планування замовлення в календарі, авторизований користувач, перебуваючи на головній сторінці додатку, натиснув на вільний часовий проміжок у календарі, заповнив форму створення замовлення, обрав клієнта, менеджера та механіка, після чого натиснув кнопку «Створити замовлення». У результаті замовлення успішно створено, воно відобразилося як у календарі, так і в загальному списку замовлень.

3. Фільтрація ресурсів у таблиці

Під час перевірки фільтрації ресурсів у таблиці, користувач відкрив сторінку «Ресурси», ввів частину назви ресурсу у поле для фільтрації та застосував фільтр. У результаті в таблиці відобразилися лише ті записи, які відповідали зазначеному фільтру, що свідчить про коректну реалізацію функціоналу.

4. Редагування замовлення

Під час редагування замовлення авторизований користувач відкрив сторінку «Замовлення», у таблиці обрав одне із замовлень та натиснув кнопку «Редагувати». Після зміни статусу і виконавця (механіка) натиснув кнопку «Зберегти». Усі зміни були успішно збережені, а також зафіксовані в історії замовлення, що підтверджує правильність функціонування.

5. Генерація акта виконаних робіт

Для перевірки генерації акта виконаних робіт було використано існуюче замовлення, до якого були додані відповідні роботи та продукти. Користувач натиснув на кнопку «Завантажити акт виконаних робіт», після чого система згенерувала та завантажила PDF-документ, що містив список робіт, використаних продуктів та загальну вартість замовлення. Всі дані були відображені коректно.

6. Оплата замовлення через QR-код

У процесі оплати замовлення через QR-код користувач обрав замовлення, доступне до оплати, та натиснув на відповідну іконку. У модальному вікні здійснив сканування QR-коду або натиснув «Перейти до оплати». Система відкрила сторінку LiqPay з попередньо заповненими даними для оплати, після чого було виконано транзакцію, збережено дані про оплату, та оновлено статус замовлення.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

7. Додавання продукту на склад

Під час додавання продукту на склад користувач відкрив сторінку «Склад», натиснув кнопку «Створити продукт», ввів необхідні дані, включаючи назву, ціну закупівлі, ціну продажу, кількість і тип продукту, після чого натиснув «Зберегти». Продукт був успішно створений, відобразився у таблиці складу з правильними даними, а також автоматично сформувалася транзакція на замовлення товару.

Висновки до розділу 3

У третьому розділі було розглянуто практичні аспекти розгортання, структури, інтерфейсу та тестування веб-орієнтованої системи для відстеження технічного обслуговування автомобілів. Насамперед описано процес запуску серверної та клієнтської частин застосунку, які реалізовані відповідно до сучасних принципів архітектури: Clean Architecture на бекенді (.NET) та SPA на фронтенді (React, Ant Design). Забезпечено інтеграцію з базою даних MS SQL через Entity Framework Core та зовнішньою платіжною системою LiqPay, що дозволяє ефективно управляти фінансовими операціями в системі.

Особливу увагу приділено інтерфейсу користувача, який є інтуїтивно зрозумілим, функціональним і адаптованим до потреб кінцевих користувачів. Завдяки використанню бібліотеки Ant Design забезпечено сучасний вигляд та зручну навігацію між модулями: замовленнями, клієнтами, співробітниками.

Було проведено ручне тестування функціональних можливостей системи. Результати тестування показали, що основні сценарії роботи – створення замовлень, авторизація користувачів, взаємодія з базою даних та платіжна інтеграція працюють стабільно, без критичних помилок. Виявлені незначні недоліки були усунуті на етапі налагодження.

Таким чином, створена система є повністю працездатною, готовою до подальшого розгортання в умовах реального використання та може слугувати як завершене рішення або основа для розширення функціоналу в межах діяльності СТО.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було комплексно вирішено завдання розробки вебзастосунку для автоматизації процесів технічного обслуговування автомобілів на СТО, що відповідає сучасним вимогам цифровізації сервісних підприємств. Проведений аналіз предметної області дозволив чітко сформулювати ключові функціональні потреби користувачів – адміністраторів, менеджерів, механіків СТО – з урахуванням реальних викликів, таких як неефективна комунікація, складне ведення обліку замовлень та потреба в контролі фінансових операцій.

Аналіз аналогічних рішень на ринку підтвердив доцільність створення адаптивного та масштабованого продукту з сучасною архітектурою, який об'єднує функції управління замовленнями, клієнтською базою, працівниками та платежами в єдиному середовищі. Обґрунтоване рішення щодо використання стеку технологій .NET (C#, Entity Framework Core) для бекенду та React з бібліотекою Ant Design для фронтенду дозволило створити гнучку структуру вебзастосунку, що взаємодіє з базою даних через RESTful API та підтримує інтеграцію з платіжними системами.

Процес проєктування забезпечив побудову структурованої архітектури програмного забезпечення з чіткою логікою доступу та ролей, а також реалізацію взаємозв'язків між сутностями бази даних. Усі основні сценарії взаємодії користувача з системою були детально змодельовані з використанням UML-діаграм, що забезпечило логічну завершеність програмної частини та зручність у подальшій підтримці продукту.

Розроблене рішення побудоване на сучасних принципах (SPA, Clean Architecture, CQRS), що забезпечило не лише функціональну завершеність, але й надійність, модульність і зручність супроводу. Забезпечено можливість подальшого розширення, зокрема, додавання нових модулів, автоматизованої звітності. Реалізація інтерфейсу з використанням Ant Design забезпечує сучасний

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

вигляд, швидку адаптацію та інтуїтивну навігацію, що важливо для щоденної роботи персоналу СТО.

Проведене ручне тестування підтвердило стабільну роботу реалізованого функціоналу в основних сценаріях від створення замовлень до обробки оплат. Усі виявлені недоліки були усунуті на етапі налагодження, що додатково засвідчує технічну готовність системи до використання в реальному середовищі.

Таким чином, результати кваліфікаційної роботи свідчать про успішну реалізацію поставленої мети, відповідність створеного вебзастосунку технічним, функціональним та експлуатаційним вимогам, а також його готовність до впровадження у виробничі процеси сучасного СТО.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке технічне обслуговування автомобіля (ТО)?. Car Service Pro. URL: <https://car-service.pro/uk/tehnicheskoe-obslyzhivanie-avtomobilya-ua>.
2. AutoRepair Cloud. Auto Repair Shop Software. URL: <https://www.autorepaircloud.com/>.
3. Програма для автосервісу та СТО. remonline.ua. URL: <https://remonline.ua/autoservice/>.
4. CRM система для простого і ефективного управління продажами. SmartCRM. URL: <https://smartcrm.binotel.ua/>.
5. Документація Shopware. URL: <https://www.shopware.com/en/>.
6. Документація React – A JavaScript library for building user interfaces. React. URL: <https://legacy.reactjs.org/>
7. Документація Foxminded. URL: <https://foxminded.ua/spa-u-prohramuvanni/>.
8. Why Single Page Application (SPA) architecture is so popular. Medium. URL: <https://medium.com/nerd-for-tech/why-single-page-application-spa-architecture-is-so-popular-141b85400204>.
9. Що таке API та кінцеві точки? Пояснення вебхуків і SDK. PayPro Global. URL: <https://payproglobal.com/uk/відповіді/що-таке-арі-та-кінцеві-точки/>
10. Переваги та недоліки .NET. DOU. Статті. URL: <https://dou.ua/lenta/articles/pros-and-cons-of-dotnet/>.
11. Документація .NET та .NET Core. Microsoft. URL: <https://learn.microsoft.com/uk-ua/lifecycle/faq/dotnet-core>.
12. Документація Ant Design 5.0. URL: <https://ant.design/components/overview/>
13. Документація Ant Design 5.0. URL: <https://ant.design/>.
14. Документація The Build Tool for the Web. Vite. URL: <https://vite.dev/>.
15. Документація Що таке .NET. EPAM Campus. URL: <https://campus.epam.ua/ua/blog/301>.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

16. Документація Entity Framework Core. Microsoft. URL: <https://learn.microsoft.com/en-us/ef/core/>.

17. Документація Entity Framework Core. Entity Framework Tutorial. URL: <https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>.

18. Документація What is Code-First?. Apidog. URL: <https://apidog.com/articles/what-is-code-first/>.

19. Документація What is SQL Server?. Microsoft. URL: <https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver17>

20. Документація What is Use Case Diagram?. Visual Paradigm. URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.

21. Документація Use Case Diagram - Unified Modeling Language (UML). Geeksforgeeks. URL: <https://www.geeksforgeeks.org/system-design/use-case-diagram/>.

22. Документація Модель діаграми зв'язків сутностей (ER). Guru99. URL: <https://www.guru99.com/uk/er-diagram-tutorial-dbms.html>

23. Документація Unified Modeling Language (UML) Diagrams. Geeksforgeeks. URL: <http://geeksforgeeks.org/unified-modeling-language-uml-introduction/>.

24. Документація State Machine Diagrams | Unified Modeling Language (UML). Geeksforgeeks. URL: <https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-state-diagrams/>.

25. Документація Sequence Diagrams - Unified Modeling Language (UML). Geeksforgeeks. URL: <https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-sequence-diagrams/>.

26. Нефункціональні вимоги: приклади, типи, підходи. Qatestlab. URL: <https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/>.

27. Advanced Design Patterns in C#: Command Query Responsibility Segregation (CQRS). Medium. URL: <https://krishan->

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

samarawickrama.medium.com/advanced-design-patterns-in-c-command-query-responsibility-segregation-cqrs-d9ff598ccdf6.

28. Clean Architecture. Medium. URL: <https://medium.com/@rudrakshnanavaty/clean-architecture-7c1b3b4cb181>.

29. Robert M. The Clean Architecture. Clean Coder. URL: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>.

30. Big calendar. Github io. URL: <https://jqense.github.io/react-big-calendar/examples/index.html?path=/story/about-big-calendar--page>.

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижевотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

Файл PasswordService

```

using GraduationWork.Application.Interfaces.Services;
using System.Security.Cryptography;
using System.Text;

namespace GraduationWork.Infrastructure.Services
{
    public class PasswordService : IPassordService
    {
        private const int KeySize = 64;
        private const int Iterations = 350000;
        private readonly HashAlgorithmName _hashAlgorithm =
            HashAlgorithmName.SHA512;

        public (string hashedPassword, string salt) HashPassword(string
            password)
        {
            var salt = RandomNumberGenerator.GetBytes(KeySize);
            var hash = Rfc2898DeriveBytes.Pbkdf2(
                Encoding.UTF8.GetBytes(password),
                salt,
                Iterations,
                _hashAlgorithm,
                KeySize
            );

            return (Convert.ToHexString(hash),
                Convert.ToBase64String(salt));
        }

        public bool VerifyPassword(string password, string hash, string
            salt)
        {
            var convertedSalt = Convert.FromBase64String(salt);
            var hashToCompare = Rfc2898DeriveBytes.Pbkdf2(password,
                convertedSalt, Iterations, _hashAlgorithm, KeySize);

            return CryptographicOperations.FixedTimeEquals(hashToCompare,
                Convert.FromHexString(hash));
        }
    }
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Файл TokenService

```

using GraduationWork.Application.Common.Options;
using GraduationWork.Application.Interfaces.Services;
using GraduationWork.Domain.Entities;
using GraduationWork.Domain.Enums;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;
using System.Text;

namespace GraduationWork.Infrastructure.Services
{
    public class TokenService : ITokenService
    {
        private readonly string _key;
        private readonly string _issuer;
        private readonly string _audience;

        public TokenService(IOptions<TokenOptions> options)
        {
            _key = options.Value.Key;
            _issuer = options.Value.Issuer;
            _audience = options.Value.Audience;
        }

        public string GenerateAccessToken(User user)
        {
            var role = user.Role!.Name;
            var permissions = RolePermissions.Map.GetValueOrDefault(role)
            ?? [];

            var claims = new List<Claim>
            {
                new (ClaimTypes.Email, user.Email),
                new (ClaimTypes.Role, role),
            };

            claims.AddRange(permissions.Select(p => new Claim("permission",
            p)));

            return GenerateToken(claims);
        }

        public string GenerateRefreshToken()
        {
            var randomNumber = new byte[64];
            using var rng = RandomNumberGenerator.Create();
            rng.GetBytes(randomNumber);

            return Convert.ToBase64String(randomNumber);
        }

        public ClaimsPrincipal GetPrincipalClaims(string token)
        {
            var tokenValidationParameters = new TokenValidationParameters

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            ValidateAudience = true,
            ValidateIssuer = true,
            ValidateIssuerSigningKey = true,
            ValidAudience = _audience,
            ValidIssuer = _issuer,
            RequireSignedTokens = false,
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_key)),
            RequireExpirationTime = false,
            ValidateLifetime = false
        };
        var tokenHandler = new JwtSecurityTokenHandler();
        var principal = tokenHandler.ValidateToken(token,
tokenValidationParameters, out var securityToken);

        var jwtSecurityToken = (JwtSecurityToken)securityToken;
        bool isAlgMatch = jwtSecurityToken.Header.Alg.Equals(
            SecurityAlgorithms.HmacSha256,
            StringComparison.InvariantCultureIgnoreCase
        );

        if (jwtSecurityToken == null || !isAlgMatch)
        {
            throw new SecurityTokenException("Invalid token");
        }
        return principal;
    }

    private string GenerateToken(IEnumerable<Claim> claims)
    {
        var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_key));
        var signingCredentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);
        var token = new JwtSecurityToken(
            _issuer,
            _audience,
            claims,
            expires: DateTime.Now.AddMinutes(180),
            signingCredentials: signingCredentials
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Файл LoginUserCommand

```

using GraduationWork.Application.Interfaces.Repositories;
using GraduationWork.Application.Interfaces.Services;
using GraduationWork.Application.Login.DTOs;
using MediatR;

namespace GraduationWork.Application.Login.Commands
{
    public record LoginUserCommand(string Email, string Password) :
    IRequest<AuthenticatedUserDTO>;

    public class LoginUserCommandHandler :
    IRequestHandler<LoginUserCommand, AuthenticatedUserDTO>
    {
        private readonly IPasswordService _passwordService;
        private readonly IUserRepository _userRepository;
        private readonly ITokenService _tokenService;

        public LoginUserCommandHandler(
            IPasswordService passwordService,
            IUserRepository userRepository,
            ITokenService tokenService
        )
        {
            _passwordService = passwordService;
            _userRepository = userRepository;
            _tokenService = tokenService;
        }

        public async Task<AuthenticatedUserDTO> Handle(LoginUserCommand
        request, CancellationToken cancellationToken)
        {
            var userByEmail = await
            _userRepository.GetUserByEmailAsync(request.Email);
            if (userByEmail is null)
                throw new Exception("User with provided email does not
            exist");
            var verifyResult =
                _passwordService.VerifyPassword(request.Password,
            userByEmail.HashedPassword, userByEmail.Salt);
            if (!verifyResult)
                throw new Exception("Login failed");

            var accessToken =
            _tokenService.GenerateAccessToken(userByEmail);
            return new AuthenticatedUserDTO
            {
                AccessToken = accessToken,
                FirstName = userByEmail.FirstName,
                LastName = userByEmail.LastName,
                RoleName = userByEmail.Role!.Name
            };
        }
    }
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Файл LiqPayController

```

using GraduationWork.Application.Common.DTOS;
using GraduationWork.Application.Interfaces.Repositories;
using GraduationWork.Application.Interfaces.Services;
using GraduationWork.Domain.Enums;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Text.Json;

namespace GraduationWork.WebApi.Controllers
{
    [ApiController]
    [Route("api/liqpay")]
    public class LiqPayController : BaseController
    {
        private readonly LiqPayService _liqPayService;
        private readonly IOrderRepository _orderRepository;
        private readonly IOrderEventsTrackingService
_orderEventsTrackingService;
        private readonly ITransactionEventsTrackingService
_transactionEventsTrackingService;

        public LiqPayController(IMediator mediator, LiqPayService
liqPayService, IOrderRepository orderRepository,
IOrderEventsTrackingService orderEventsTrackingService,
ITransactionEventsTrackingService transactionEventsTrackingService) :
base(mediator)
        {
            _liqPayService = liqPayService;
            _orderRepository = orderRepository;
            _orderEventsTrackingService = orderEventsTrackingService;
            _transactionEventsTrackingService =
transactionEventsTrackingService;
        }

        [HttpPost("payment/generate")]
        [AllowAnonymous]
        public async Task<IActionResult> GeneratePayment([FromBody]
LiqPayRequestDto request, CancellationToken cancellationToken)
        {
            if (!Guid.TryParse(request.OrderId, out var orderId))
                return BadRequest("Invalid OrderId format");

            var result = await _liqPayService.GeneratePaymentData(orderId,
request.PaymentSource, cancellationToken);
            var data = result.data;
            var signature = result.signature;

            var paymentUrl =
$"https://www.liqpay.ua/api/3/checkout?data={data}&signature={signature}";

            return Ok(new { data, signature, paymentUrl });
        }
    }
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

[HttpPost("payment/callback")]
[AllowAnonymous]
public async Task<IActionResult> PaymentCallback([FromForm]
IFormCollection form, CancellationToken cancellationToken)
{
    var data = form["data"];
    var signature = form["signature"];

    if (string.IsNullOrEmpty(data) ||
string.IsNullOrEmpty(signature))
    {
        return BadRequest("Missing data or signature");
    }

    if (!_liqPayService.IsValidSignature(data, signature))
    {
        return BadRequest("Invalid signature");
    }

    var payload = LiqPayService.DecodeData(data);

    var orderIdRaw = payload.ContainsKey("order_id") ?
payload["order_id"]?.ToString() : null;
    var status = payload.ContainsKey("status") ?
payload["status"]?.ToString() : null;

    if (string.IsNullOrEmpty(orderIdRaw) ||
string.IsNullOrEmpty(status))
    {
        return BadRequest("Missing required payment information.");
    }

    if (!Guid.TryParse(orderIdRaw, out var orderId))
    {
        return BadRequest("Invalid order_id format.");
    }

    if (status == "success")
    {
        var order = await _orderRepository.GetByIdAsync(orderId,
cancellationToken);

        if (order == null)
        {
            return NotFound($"Order with ID {orderId} not found.");
        }

        if (order.Status != OrderStatus.Closed)
        {
            order.Status = OrderStatus.Closed;
            order.CompletedDate = DateTime.UtcNow;

            await _orderRepository.CreateOrUpdateAsync(order,
cancellationToken);

            await _orderEventsTrackingService.AddHistoryEvent(
                order.Id,

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        OrderEventType.PaymentSucceeded,
        "liqpay-system",
        new
        {
            PaymentStatus = "success",
            PaymentTime = DateTime.UtcNow
        });

        var amountElement = (JsonElement)payload["amount"];
        decimal amount = amountElement.ValueKind ==
JsonValueKind.Number
        ? amountElement.GetDecimal()
        : decimal.Parse(amountElement.GetString());

        await
        _transactionEventsTrackingService.AddHistoryEvent(
            eventType:
TransactionEventType.PaymentSucceeded.ToString(),
            details: new Dictionary<string, object>
            {
                { "OrderNumber", order.OrderNumber },
                { "Amount", amount }
            });
    }

    return Ok();
}
}
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Файл OrderEventsTrackingService

```

using GraduationWork.Application.Interfaces.Repositories;
using GraduationWork.Application.Interfaces.Services;
using GraduationWork.Application.OrderHistoryEvents;
using GraduationWork.Application.Orders.DTOs;
using GraduationWork.Domain.Entities;
using GraduationWork.Domain.Enums;
using System.ComponentModel.DataAnnotations;
using System.Text.Json;

namespace GraduationWork.Infrastructure.Services
{
    public class OrderEventsTrackingService : IOrderEventsTrackingService
    {
        private readonly IOrderHistoryEventRepository
        _orderHistoryEventRepository;
        private readonly IUserRepository _userRepository;
        private readonly ICustomerRepository _customerRepository;
        private readonly IResourcesRepository _resourcesRepository;

        public OrderEventsTrackingService(
            IOrderHistoryEventRepository orderHistoryEventRepository,
            IUserRepository userRepository,
            ICustomerRepository customerRepository,
            IResourcesRepository resourcesRepository)
        {
            _orderHistoryEventRepository = orderHistoryEventRepository;
            _userRepository = userRepository;
            _customerRepository = customerRepository;
            _resourcesRepository = resourcesRepository;
        }

        private string GetEnumDisplayName(Enum value)
        {
            var fieldInfo = value.GetType().GetField(value.ToString());
            var attribute =
            (DisplayAttribute)Attribute.GetCustomAttribute(fieldInfo,
            typeof(DisplayAttribute));
            return attribute?.Name ?? value.ToString();
        }

        public async Task<List<OrderHistoryEventDto>>
        ProcessOrderEventsAsync(List<OrderHistoryEvent> orderHistoryEvents)
        {
            var resultList = new List<OrderHistoryEventDto>();

            foreach (var orderHistoryEvent in orderHistoryEvents)
            {
                if (orderHistoryEvent.EventType == OrderEventType.Created)
                {
                    resultList.Add(new OrderHistoryEventDto
                    {
                        Description = "Замовлення створено",
                        Type = orderHistoryEvent.EventType,
                    });
                }
            }
        }
    }
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

    }

    if (orderHistoryEvent.EventType ==
OrderEventType.StatusChanged)
    {
        var statusChangedDetails =
JsonSerializer.Deserialize<StatusChangedDto>(orderHistoryEvent.DetailsJson)
;

        OrderStatus fromStatusEnum =
(OrderStatus)Enum.Parse(typeof(OrderStatus),
statusChangedDetails.FromStatus.ToString());
        OrderStatus toStatusEnum =
(OrderStatus)Enum.Parse(typeof(OrderStatus),
statusChangedDetails.ToStatus.ToString());
        string fromStatusDisplayName =
GetEnumDisplayName(fromStatusEnum);
        string toStatusDisplayName =
GetEnumDisplayName(toStatusEnum);

        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Статус змінено з
{fromStatusDisplayName} на {toStatusDisplayName}"
        });
    }

    if (orderHistoryEvent.EventType ==
OrderEventType.ManagerChanged)
    {
        var managerChangedDetails =
JsonSerializer.Deserialize<ManagerChangedDto>(orderHistoryEvent.DetailsJson)
);
        var users = await
_userRepository.GetUsersByIdsAsync(new List<Guid>())
        {
            Guid.Parse(managerChangedDetails.ToManager),
            Guid.Parse(managerChangedDetails.FromManager)
        }, CancellationToken.None);

        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Манеджера змінено з
{users[1].FirstName} {users[1].LastName} на {users[0].FirstName}
{users[0].LastName}"
        });
    }

    if (orderHistoryEvent.EventType ==
OrderEventType.ExecutorChanged)
    {
        var managerChangedDetails =
JsonSerializer.Deserialize<ExecutorChangedDto>(orderHistoryEvent.DetailsJso
n);
        var users = await
_userRepository.GetUsersByIdsAsync(new List<Guid>())

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            Guid.Parse(managerChangedDetails.ToExecutor),
            Guid.Parse(managerChangedDetails.FromExecutor)
        }, CancellationToken.None);
        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Механіка змінено з
{users[1].FirstName} {users[1].LastName} на {users[0].FirstName}
{users[0].LastName}"
        });
    }
    if (orderHistoryEvent.EventType ==
OrderEventType.CustomerChanged)
    {
        var managerChangedDetails =
JsonSerializer.Deserialize<CustomerChangedDto>(orderHistoryEvent.DetailsJso
n);
        var toCustomer = await
_customerRepository.GetByIdAsync(Guid.Parse(managerChangedDetails.ToCustome
r));
        var fromCustomer = await
_customerRepository.GetByIdAsync(Guid.Parse(managerChangedDetails.FromCusto
mer));
        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Клієнта змінено з
{fromCustomer.FirstName} {fromCustomer.LastName} на {toCustomer.FirstName}
{toCustomer.LastName}"
        });
    }
    if (orderHistoryEvent.EventType ==
OrderEventType.ResourceChanged)
    {
        var managerChangedDetails =
JsonSerializer.Deserialize<ResourceChangedDto>(orderHistoryEvent.DetailsJso
n);
        var toResource = await
_resourcesRepository.GetByIdAsync(Guid.Parse(managerChangedDetails.ToResour
ce));
        var fromResource = await
_resourcesRepository.GetByIdAsync(Guid.Parse(managerChangedDetails.FromReso
urce));
        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Ресурс змінено з
{fromResource.Name} на {toResource.Name}"
        });
    }
    if (orderHistoryEvent.EventType ==
OrderEventType.CompletedWorkAdded)
    {
        var details =
JsonSerializer.Deserialize<Dictionary<string,
object>>(orderHistoryEvent.DetailsJson);

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Додано виконану роботу:
{details["WorkName"]} (К-сть: {details["Quantity"]})"
        });
    }
    if (orderHistoryEvent.EventType ==
OrderEventType.ProductAdded)
    {
        var details =
JsonSerializer.Deserialize<Dictionary<string,
object>>(orderHistoryEvent.DetailsJson);
        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Додано продукт:
{details["ProductName"]} (К-сть: {details["Quantity"]})"
        });
    }
    if (orderHistoryEvent.EventType ==
OrderEventType.PaymentSucceeded)
    {
        resultList.Add(new OrderHistoryEventDto
        {
            Type = orderHistoryEvent.EventType,
            Description = $"Замовлення оплачено"
        });
    }
}

return resultList;
}

public async Task TrackOrderHistoryAsync(CreateOrUpdateOrderDto
newVersion, Order oldVersion)
{
    if (newVersion.ManagerId != oldVersion.ManagerId)
    {
        await AddHistoryEvent(
            oldVersion.Id,
            OrderEventType.ManagerChanged,
            newVersion.ManagerId.ToString(),
            new
            {
                FromManager = oldVersion.ManagerId,
                ToManager = newVersion.ManagerId,
            }
        );
    }

    if (newVersion.ExecutorId != oldVersion.ExecutorId)
    {
        await AddHistoryEvent(
            oldVersion.Id,
            OrderEventType.ExecutorChanged,
            newVersion.ManagerId.ToString(),
            new
            {

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижевотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

        FromExecutor = oldVersion.ExecutorId,
        ToExecutor = newVersion.ExecutorId,
    }
    );
}

if (newVersion.CustomerId != oldVersion.CustomerId)
{
    await AddHistoryEvent(
        oldVersion.Id,
        OrderEventType.CustomerChanged,
        newVersion.ManagerId.ToString(),
        new
        {
            FromCustomer = oldVersion.CustomerId,
            ToCustomer = newVersion.CustomerId,
        }
    );
}

if (newVersion.ResourceId != oldVersion.ResourceId)
{
    await AddHistoryEvent(
        oldVersion.Id,
        OrderEventType.ResourceChanged,
        newVersion.ManagerId.ToString(),
        new
        {
            FromResource = oldVersion.ResourceId,
            ToResource = newVersion.ResourceId,
        }
    );
}

if (newVersion.Status != oldVersion.Status)
{
    await AddHistoryEvent(
        oldVersion.Id,
        OrderEventType.StatusChanged,
        newVersion.ManagerId.ToString(),
        new
        {
            FromStatus = oldVersion.Status,
            ToStatus = newVersion.Status,
        }
    );
}
}

public async Task AddHistoryEvent(Guid orderId, string eventType,
string changedBy, object details)
{
    var historyEvent = new OrderHistoryEvent
    {
        Id = Guid.NewGuid(),
        OrderId = orderId,
        EventType = eventType,

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижевотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        ChangedBy = changedBy,
        Timestamp = DateTime.UtcNow,
        DetailsJson = JsonSerializer.Serialize(details)
    };

    await
_orderHistoryEventRepository.CreateOrUpdateAsync(historyEvent,
CancellationTokens.None);
    }
}
}

```

		Миронова В.Л.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81