МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна система надання послуг з ремонту автомобілів»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Ободяк В. К.**

Студент групи ІН.м-92 — **Приказчик А.О.**

СУМИ 2020

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Приказчику Андрію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна система надання послуг з ремонту автомобілів*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Огляд існуючих рішень. 2) Постановка задачі. 3) Вибір засобів реалізації. 4) Моделювання та проектування системи. 5) Практична реалізація.*

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Огляд існуючих аналогів*	*01.09 – 12.09*	
2.	*Постановка завдання та вибір методів реалізації*	*15.09 – 26.09*	
3.	*Моделювання та проектування web-додатку.*	*29.09 – 30.10*	
4.	*Практична реалізація.*	*03.11 – 28.11*	
5.	*Оформлення пояснювальної записки до дипломної роботи*	*01.12 – 10.12*	

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна система надання послуг з ремонту автомобілів».

Метою даної роботи є розробка інформаційної системи для замовників та сервісів чи майстрів у сфері обслуговуванні автомобілів.

Пояснювальна записка містить 81 сторінку, 7 таблиць, 39 рисунків, 22 джерела, 1 додаток.

В першому розділі проведено аналіз обраної предметної області. У цьому розділі виконано: огляд останніх актуальних досліджень та сервісів, аналіз аналогів.

В другому розділі було сформовано головну мету та задачі дослідження. Даний розділ включає список основних задач для створення інформаційної системи, а також вибір засобів реалізації.

В третьому розділі описано проектування системи надання послуг з ремонту автомобілів. Було виконано IDEF0 та IDEF1 діаграми варіантів використання функціоналу інформаційної системи. Також спроектовано базу даних.

В четвертому розділі було виконано реалізацію та детальний опис використання інформаційної системи зі сторони власника автомобіля, механіка та автомобільного сервісу. Додатково описані можливості адміністратора.

Результатом проведеної роботи є розроблена інформаційна система надання послуг з ремонту автомобілів.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, АВТОСЕРВІС, ЗАМОВЛЕННЯ, JAVASCRIPT, PHP, LARAVEL, ФРЕЙМВОРК.

ЗМІСТ

ВСТУП

Прогрес не стоїть на місці і з кожним роком випускаються все більш складні автомобілі. Неможливо знайти фахівця, який розбирається в автомобілях всіх марок і моделей. А покладатися на власні сили в ремонту авто – більш ніж нерозумно. Для вирішення цього складного завдання існують сервіси технічного обслуговування, які використовують сучасне діагностичне обладнання. Крім того обслуговування виконується висококваліфікованими працівниками.

Регулярне технічне обслуговування – найважливіше правило змісту будь-якого сучасного авто. Періодична заміна деталей, фільтрів, перевірка сходження і підвіски, ходової, вузлів силового агрегату і навіть підкачка шин – все це необхідно виконувати на перевіреному автосервісі.

Проте станції техобслуговування, де надають якісний сервіс і ремонт автомобіля, можуть грішить халтурою.

Для швидкого вирішення проблем, виявлених в ході досліження, було вирішено розробити сайт для надання зручного пошуку сервісів технічного обслуговування, щоб кожен бажаючий міг з легкістю знайти бажаний сервіс і записатись на обслуговування. Для реалізації поставленої мети потрібно вирішити такі задачі:

- Дослідити процеси роботи СТО.
- Провести аналіз аналогів сайтів пошуку.
- Визначити задачі сайту.
- Обрати засоби та провести реалізацію.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Більше 60% автолюбителів в умовах кризи відмовилися від послуг, які пропонують автомайстерні. Самостійний ремонт автомобіля – не найгірший варіант економії грошей, якщо власник розбирається в техніці, або має знайомого автослюсаря.

Переваги проведення ремонтних робіт своїми руками:

– Безкоштовна робота в зручний час.

– Зниження витрат на покупку витратних матеріалів, запчастин та інше.

– Постійний контроль стану машини.

– Мінуси і самостійна заміна деталей.

– Зняття з гарантії в дилерському центрі, якщо автомобіль новий (незалежно від віку і пробігу).

– Низька якість роботи (кваліфікований фахівець виконає роботу краще, ніж недосвідчений любитель).

– Незначні на перший погляд помилки можуть привести до серйозних поломок або повного виходу з ладу транспортного засобу.

– Витрати на придбання спеціальних інструментів.

Більшість автовласників намагаються обслуговуватися на одному і тому ж автосервісі, якість якого їх влаштовує. Але іноді виникає необхідність в ремонті, яким це СТО не займається. Буває так, що персонал на сервісі змінюється, і якість їх роботи вже не влаштовує. Виникає необхідність в пошуку нового сервісу технічного обслуговування [1].

Виконання простого ремонту вдома може бути навіть корисним: автовласник буде в курсі будови машини, заощадить на усуненні незначних поломок і покупці дорогих запчастин.

Проте професіонали не рекомендують самостійно виконувати складні кузовні роботи, що вимагають знань і досвіду, а також застосування спеціальних інструментів. Імпортні дорогі машини краще обслуговувати по гарантії в дилерських центрах. Додаткова перевага полягає в тому, що автолюбителю не потрібно буде шукати запчастини, переплачувати за терміновість доставки. Це дозволить заощадити час, нерви і гроші.

Тому більшість автовласників намагаються обслуговуватися в автосервісі, який їх влаштовує. Але іноді виникає необхідність в ремонті, яким це СТО не займається. Буває так, що персонал на сервісі змінюється, і якість їх роботи вже не влаштовує [2]. Виникає необхідність в пошуку нового сервісу технічного обслуговування.

Щоб підібрати хороший сервіс необхідно слідкувати певним критеріям які наведені в таблиці 1.1.

Таблиця 1.1 – Критерії сервісів

Характеристика	Опис
Зовнішній вигляд	Чисті зовні і всередині бокси, охайні співробітники, порядок на робочих місцях – все це вимагає інвестицій, і якщо власник не поскупився на них, значить він серйозно ставиться не тільки до подібних «дрібниць», а й до найважливіших питань роботи підприємства. Крім того, згадані зовнішні ознаки – свідчення рівня дисципліни в колективі.
Діалог з клієнтом	Чи є на СТО комп'ютерна база клієнтів, пропонуються їм варіанти при підборі запчастин і розрахунках за виконану роботу, чи можна розплатитися через банк? Підприємство, яке дорожить своїм ім'ям і відповідає за свою роботу, завжди прагне хоч у чомусь піти назустріч клієнтові.

Продовження таблиці 1.1

Характеристика	Опис
Умови гарантії	На жаль, повноцінна гарантія на виконаний автосервісом ремонт або техобслуговування в наш час все ще рідкість. Навіть на багатьох серйозних СТО гарантійні зобов'язання бувають «розмиті» цілою низкою умов і застережень. Але тут важливий вже сам факт – чи готова майстерня відповідати за виконану роботу? Тому потрібно або погоджуватися на надання запчастин майстерні, або погоджувати умови гарантії на одну тільки роботу.
Допуск клієнта в ремзону	Адміністрація, яка дозволяє власнику машини спостерігати за ремонтом, впевнена в своїх співробітниках і не боїться, що власник автомобіля побачить щось зайве. Це з великим ступенем ймовірності підтверджує високу якість ремонту. Варіантів може бути два: клієнт спостерігає за роботою з його машиною через скляну стіну спеціальної кімнати очікування або зайшовши безпосередньо в цех. У другому випадку для забезпечення безпеки клієнт проходить короткий інструктаж і одягається в каску і яскравий одяг.
Персонал	В Україні багато постачальників запчастин і ремонтного обладнання проводять свої власні семінари для підприємств-партнерів, в тому числі для механіків СТО. Дипломи про закінчення курсів автомеханіка багато чого варті (в тому числі і буквально), оскільки, з одного боку, означають, що власник майстерні не шкодує грошей ні на

Продовження таблиці 1.1

Характеристика	Опис
Персонал	підготовку свого персоналу, ні на закупівлю якісного оснащення і запчастин. А з іншого – факт навчання говорить про певний рівень визнання станції.
Відкритість	Наявність інформації про сервіс технічного обслуговування в Інтернеті (на незалежних форумах). Звертайте увагу на авторів відгуків, вони повинні бути реальними людьми з аккаунтами і з історією в соціальних мережах. Також важлива участь СТО в будь-яких рейтингах, конкурсах майстрів, професійних змаганнях і т.д. Відкрите життя підприємства стане хорошим свідченням його надійності, адже виходить, що за його роботу перед нами ручаються авторитетні комісії, що складаються з професійних експертів.

Тому на вибір СТО краще витратити чимало зусиль, адже їх кількість в сьогодні дуже велика. Вклавши більше часу на пошуку сервісу, автовласник істотно заощадить на ремонті авто [3].

Отже для полегшення пошуку бажаного сервісу технічного обслуговування було вирішено розробити інформаційну систему яка б надавала можливість з легкістю знайти необхідний сервіс і надавати детальну інформацію про нього.

1.2 Аналіз аналогів

Процес розробки нового продукту неможливий без попередньої постановки завдань і цілей а також ретельного аналізу ринкової ніші. Аналіз web-сайтів конкурентів, аналіз цільової аудиторії майбутнього сайту, виявлення «сильних» і «слабких» сторін проекту. Розробка комерційного сайту завжди починається з виконання подібних завдань [4]. Перед початком розробки поставленої задачі потрібно провести аналіз існуючих аналогів.

Для досягнення мети було проведено дослідження таких веб-сайтів:

– carbook.ua.
– vse-sto.com.ua.
– autobooking.com.

Одним із ресурсів з подібною тематикою і призначення являється сайт carbook.ua призначений для пошуку центрів сервісного обслуговування, автомобільних мийок та автостанцій. Веб-сайт має досить зручний і сучасний інтерфейс, зручний і зрозумілий пошук (рис. 1.1).

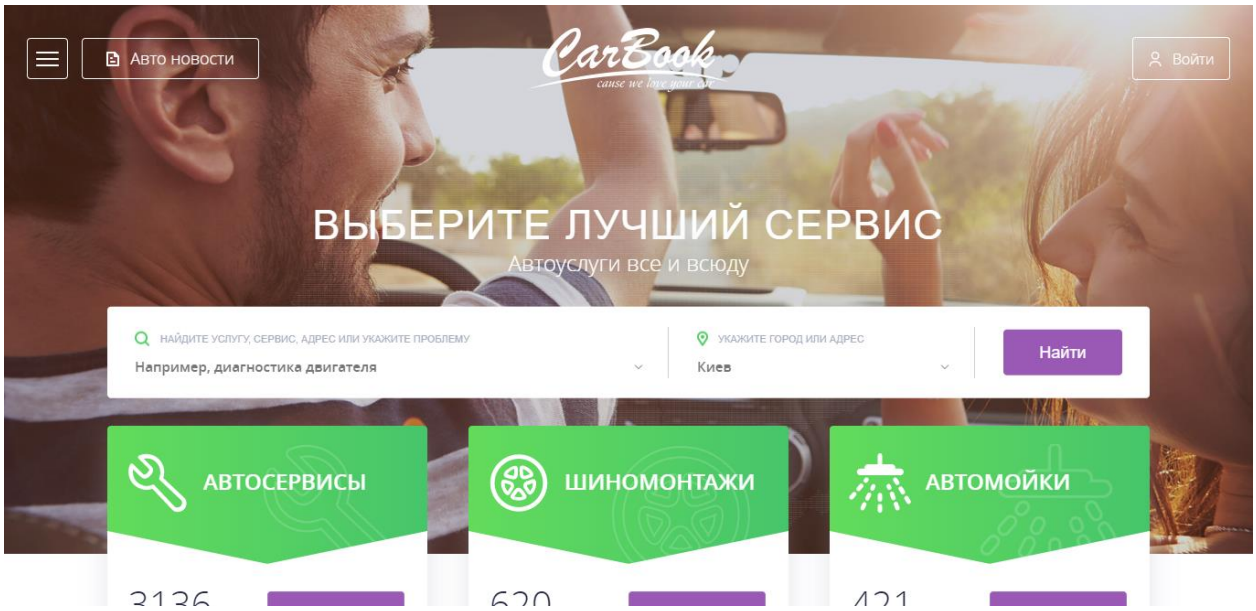

Рисунок 1.1 – Головна сторінка carbook.ua

До переваг даного сайту можна віднести зручний інтерфейс, актуальну інформацію, зручний пошук з можливістю фільтрації по відповідним критеріям, наявність детальної інформації про сервіс та види послуг [5]. Також на даному сайті постійно публікуються автомобільні новини.

Проте представлений сайт має декілька недоліків. До них можна віднести відсутність калькулятора цін послуг, що є критичним для користувача. Також на сайті відсутня можливість надання персональних послуг з ремонту авто. Крім того відсутня можливість написати повідомлення власнику СТО.

Наступним сайтом для аналізу було обрано каталог СТО міста Київ – vse-sto.com.ua (рис. 1.2).

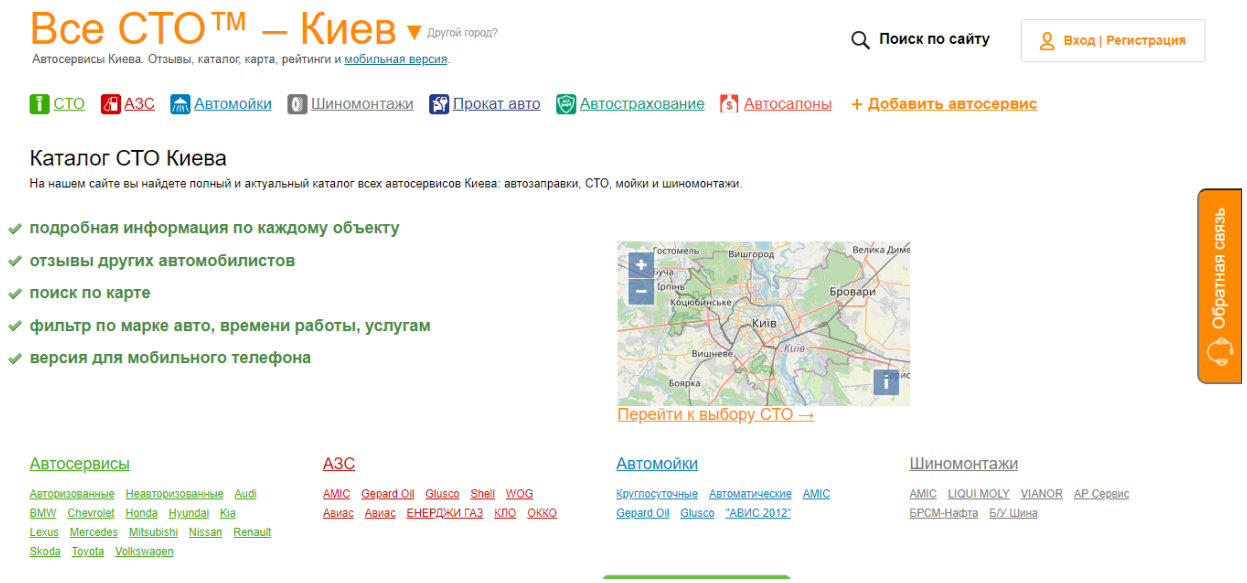

Рисунок 1.2 – Головна сторінка vse-sto.com.ua

Даний сайт має декілька переваг, а саме можливість зворотного зв’язку з адміністрацією та актуальну інформацію. Проте представлений сайт має досить велику кількість недоліків. До них можна віднести застарілий дизайн, що є критичним для користувача. Відсутність зручної панелі навігації, відсутність пошуку сервісів з можливістю фільтрації, а також відсутність калькулятору ціни ремонту авто.

Для аналізу також було обрано сайт пошуку СТО autobooking.com головна сторінка якого зображена на рисунку 1.3.

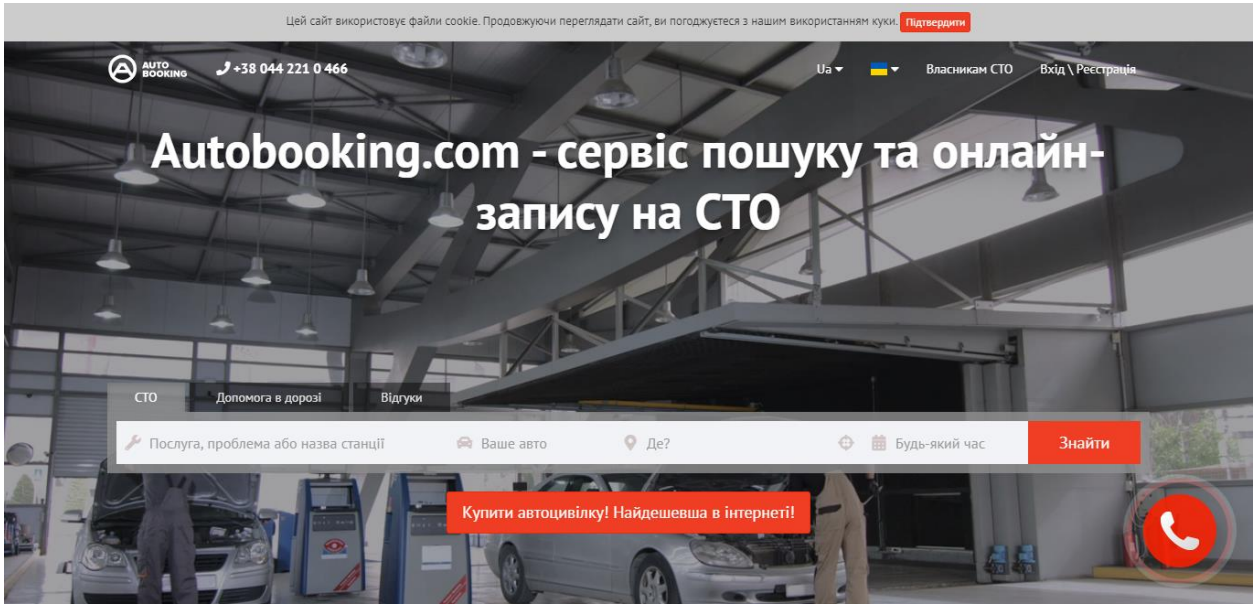

Рисунок 1.3 – Головна сторінка autobooking.com

Даний сайт вирізняється сучасним дизайном, зручним пошуком сервісів технічного обслуговування, можливістю записатись на обслуговування та наявність зворотного зв’язку з адміністрацією сайту.

Проте даний сайт також має ряд недоліків. А саме відсутність детальної інформації про сервіс, калькулятора цін на послуги та відсутність можливості написати повідомлення власнику СТО [6].

Таблиця 1.2 – Аналіз розглянутих сайтів

Критерії	carbook.ua	vse-sto.com.ua	autobooking.com
Відсутність непотрібної реклами	+	+	+

Продовження таблиці 1.2

Критерії	carbook.ua	vse-sto.com.ua	autobooking.com
Неактуальна інформація	-	-	-
Можливість зворотного зв'язку	-	+	-
Наявність калькулятора	-	-	-
Повільна робота сайту	-	+	+
Застарілий дизайн	-	+	-

Отже, в результаті проведеного аналізу сайтів зі схожою тематикою і призначенням було вирішено розробити сайт який не матиме недоліків, виявлених на сайтах представлених в таблиці 1.2.

Дана розробка матиме всі переваги даних сайтів, а також матиме унікальний дизайн та зручний інтерфейс [7].

2 ПОСТАНОВКА ЗАДАЧІ І МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Постановка задачі

Виходячи із визначеної вище актуальності проблеми, було визначено, що потрібно створити веб-додаток для того, щоб кожен бажаючий міг з легкістю знайти сервіс ремонту авто відповідно його вимогам, а також відразу підрахувати ціну ремонту. Основною аудиторією сайту будуть користувачі з базовими навичками користування комп’ютером.

Проект повинен бути реалізований у вигляді веб-додатку, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями [8].

Інформаційна система повинна мати наступні можливості:

– Пошук СТО з можливістю фільтрації по моделі авто та виду поломки.

– Можливість автоматично підраховувати ціну ремонту.

– Перегляд повної інформації про сервіс.

– Можливість реєстрації та авторизації користувачів.

– Залишати відгуки.

– Записуватись на ремонт.

Також ІС повинна відповідати вимогам отриманим у технічному завданні, а також має бути утверджена замовником.

Веб-додаток, повинен володіти наступними особливостями:

– Сайт повинен бути повністю функціональний, а також мати інтуїтивно зрозумілий інтерфейс.

– Веб-сервіс повинен підтримувати графічні вставки, та анімації.

Основним завданням проектування є створення такої системи яка б дозволяла за різними характерами поломки швидко знайти сервіс технічного

обслуговування, відразу підрахувати ціну ремонту авто та зареєструвати заявку на ремонт [9].

Для реалізації поставленої мети потрібно вирішити такі задачі:

1. Постановка цілей і завдань сайту.
2. Створення та опрацювання технічного завдання (ТЗ) на розробку сайту.
3. Вивчити процеси пошуку СТО.
4. Обрати та налаштувати інструменти реалізації.
5. Проектування та розробка сайту.
6. Наповнення контентом.
7. Тестування сайту та розміщення в Інтернет мережі.

Детальна інформація щодо розробки веб-додатку наведена у розробленому в ході аналізу предметної області технічному завданні (Додаток А).

2.2 Вибір засобів реалізації

Розробка веб-сайтів – це процес створення веб-сайтів та додатків для інтернету або для приватної мережі, відомий як інтернет.

Будь-який сайт складається з призначеної для користувача і серверної частин. На сторінці в інтернеті ви бачите текст, кнопки, панелі, зображення і відео. Можете переміщатися по сайту, вільно вивчати контент. Як відомо, за візуалізацію, інтерактивність і зрозумілість інтерфейсу відповідає frontend-розробник. Користувач бачить красивий дизайн, підсвічені кнопки та цікаву типографіку, сайтом зручно користуватися.

Під час розробки інтерфейсу було використано відому мову розмітки HTML для створення каркасу сторінок сайту та мову каскадних таблиць стилів, відому як CSS, для присвоєння стилю блокам та сторінкам, а також мову

програмування JavaScript для надання сайту певного функціоналу та інтерактивності.

Сучасні тенденції розробки інтерфейсу звичайно вимагають використання JavaScript фреймворків, щоб прискорити та полегшити розробку. Далі на графіку наведено загальний рейтинг найбільш популярних фреймворків 2020 року (рис. 2.1).

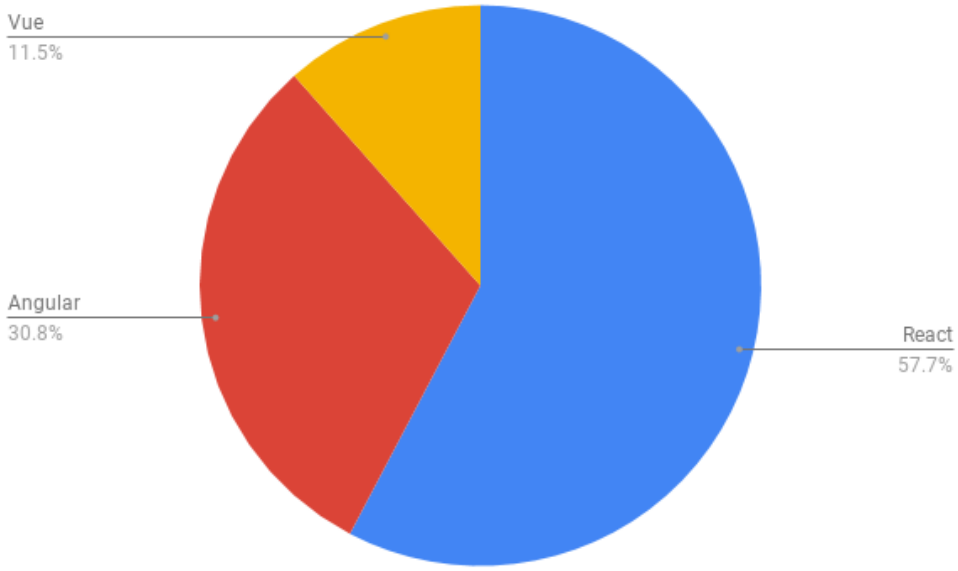

Рисунок 2.1 – Графік популярності javascript-фреймворків

React – це ефективна та відома гнучка декларативна бібліотека JavaScript, що використовується для збірки UI. Розроблена дана бібліотека була командою із Facebook. Даний фреймворк дозволяє без зусиль створювати інтерактивний інтерфейс. Адже, React не просто підтримує збірку об'єктно-центричних додатків. Крім того, розробники даного фреймворку з усією серйозністю поставилися до зворотної сумісності, так що в довговічності свого застосування можете бути впевнені. За графіком зображеним вище добре видно, що за останні кілька років рівень обізнаності про React значно підвищився [10].

Vue.js – прогресивний фреймворк у 2020 році призначений для збірки інтерфейсів користувача. Він був розроблений Еваном Ю та ще 234 асистентами фахівця, набрав більше ніж 121 тисячі «зірок» на платформі GitHub. Даний фреймворк включає доступну кореневу бібліотеку, яка у першу

чергу вирішує завдання рівня уявлення й екосистему деяких додаткових бібліотек. Це дозволяє створювати складні та об'ємні Single-Page сторінки.

Наступний фреймворк – Angular. Angular розроблений компанією Google, що вже отримав більше ніж 44 тисячі «зірок» на GitHub. Він являє з себе платформу, яка значно спрощує збірку додатків в інтернет. Це не дивно, адже Angular поєднує декларативні шаблони та впровадження залежності, та двостороннє зв'язування даних, та кращі практики вирішення проблем розробки. Фреймворк дозволяє збирати додатки для веб-систем, мобільних пристроїв та більшості настільних ПК. У ній пропонується самий зручний та зрозумілий для більшості початківців інтерфейс командного рядка чи консоль [11].

За логіку, працездатність і правильне функціонування сайту відповідає серверна частина, яка прихована від користувача. Її створенням займається backend-розробник, а управляти може лише адміністратор сайту через спеціальний інтерфейс.

Будь-який запит, який робить користувач, передається на сервер. Вся робота відбувається там [12]. Запит обробляється, фільтрується, а відповідь відправляється назад. Backend-розробка відповідає за правильне виконання цього процесу.

Як відомо, на сьогоднішній день PHP – найпоширеніша мова веб-програмування. Переважна більшість сайтів та веб-сервісів в інтернеті написано саме за допомогою PHP. За деякими оцінками, дана мова програмування застосовується більш ніж на 81% сайтів, серед яких навіть такі системи, як facebook, badoo та інші. Популярність PHP обумовлена такими факторами як простота мови, яка дозволяє швидко і легко створювати сайти і портали різної складності. PHP був створений в 1994 році програмістом Расмусом Лердорфом та спочатку був набір скриптів на іншій мові Perl [13].

Як відомо, пізніше цей набір скриптів був переписаний в інтерпретатор на мові Сі. І з самого виникнення PHP представляв зручний набір інструментів для спрощеного створення веб-сайтів і веб-додатків.

Переваги PHP:

- Використовується для поширених операційних систем, таких як Windows, Mac OS, Linux та власні версії пакетів розробки на PHP, а це значить, що можна створювати веб-сервіси на будь-які з даних операційних систем.
- PHP може працювати в зв'язці з різними іншими веб-серверами, такими як Apache, Nginx, IIS.
- Значна простота та легкість в освоєнні.
- Схожий синтаксис із мовою С.
- Підтримує роботу із значною кількістю систем баз даних. Наприклад: mysql, MSSQL, Oracle, Postgre, mongodb та інші.
- Поширеність послуг хостингу.
- Постійний розвиток.

Проте для спрощення розробки і підвищення ефективності були створені фреймворки.

Фреймворк є перш за все каркасом, призначеним для створення динамічних веб-сторінок, веб-додатків, служб або інших ресурсів. Він спрощує розробку і позбавляє від необхідності писати звичайний код. Фреймворк спрощує доступ до бази даних, розробку інтерфейсу та зменшує дублювання коду. Розглянемо один з популярних фреймворків Yii. Він є універсальним фреймворком та може бути задіяний у всіх типах веб-додатків.

Завдяки своїй складовій структурі та чудовій підтримці кешу, фреймворки особливо підходять для розробки великих проектів, таких як портали, форуми, CMS, магазини або програми RESTful. Yii – це командний проект. Він підтримується і розвивається сильною командою та великою спільнотою розробників. Автор цього фреймворку стежать за тенденціями розвитку веб-сайтів та інших проектів.

Найбільш підходящі можливості та кращі практики регулярно впроваджуються в фреймворк в вигляді простих й елегантних інтерфейсів:

– Переважно використовує архітектура MVC.

– Дотримується філософії простого і елегантного коду не намагаючись ускладнювати дизайн тільки заради проходження будь-якими шаблонами проектування.

– Фреймворком є full-stack та включає в собі перевірені і добре зарекомендовані в собі можливості: ActiveRecord для реляційних і NoSQL баз даних, підтримку REST API, багаторівневе кешування та інше.

– Можна налаштувати чи замінити практично будь-яку частину основного коду. Також легко ділитися кодом та використовувати код спільнот [14].

Розглянемо також не менш популярний фреймворк Laravel. Даний фреймворк використовується для веб-додатків із виразним й елегантним синтаксисом. Він дозволить спростити вирішення основних завдань, таких як аутентифікація, маршрутизація, сесії чи кешування. Laravel – це спроба об'єднати найкраще, що використовується в інших фреймворків, а також Ruby on Rails, ASP.NET та Sinatra [15].

Laravel звичайно доступний, але й потужний. Він має безліч відмінних інструментів для великих й надійних додатків:

– Для організації коду використовує архітектурний MVC.

– Функціонал Laravel є простим для розуміння і використання.

– Більшість функцій чудово працюють та спираються на загальноприйняті стандарти написання коду.

– Документація зручна й постійно оновлюється.

– Інтегрована система модульного тестування.

На основі проведеного аналізу був зроблений вибір на користь Laravel, який зараз активно розвивається, його структура дозволяє легко масштабувати наш додаток. Yii є більш стабільним, але менш масштабованим.

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1 Структурно-функціональне моделювання процесу

IDEF0 – це багато в чому дуже простий метод. Кожне поле представляє окремий процес, як і в інших підходах, але IDEF0 відрізняється використанням та розміщенням стрілок [18]. Крім звичайних входів і виходів, існують ще два типи стрілок, які представляють "елементи управління" та "механізми".

Перелік головних функцій даної інформаційної системи:

1. Реєстрація на сайті.
2. Авторизація на сайті.
3. Налаштування особистого кабінету.
4. Написання повідомлення адміністратору.
5. Оформлення розділу «Навички».
6. Перегляд аккаунтів СТО чи окремих виконавців.
7. Перегляд контактної інформації.
8. Використання вбудованого калькулятора для обрахування ціни ремонту автомобіля.
9. Оформлення замовлення послуги або консультації.
10. Написання відгуку.
11. Можливість листування користувача інформаційної системи (СТО/інші виконавці).

Елементи управління є формою введення, але які використовуються для керування діяльністю в процесі. Іноді виникає певна міра невизначеності щодо того, є елемент елементом введення чи контролю. Розглянемо IDEF0 інформаційної системи (рис. 3.1).

Рисунок 3.1 – IDEF0 інформаційної системи

Методологія IDEF1 розділяє елементів структури інформаційної галузі, їх властивості та взаємозв'язки на класи. Центральним поняттям методології IDEF1 є поняття сутності. Клас сутностей являє собою сукупність інформації, накопиченої і зберігається в рамках підприємства і відповідає певному об'єкту або групі об'єктів реального світу [19]. Основними концептуальними властивостями сутностей в IDEF1 є:

– Стійкість. Інформація, що має відношення до тієї чи іншої суті постійно накопичується.

– Унікальність. Будь-яка сутність може бути однозначно ідентифікована з іншої сутності.

Кожна сутність має своє ім'я і атрибути. Атрибути представляють собою характерні властивості і ознаки об'єктів реального світу, які стосуються певної сутності. Клас атрибутів являє собою набір пар, що складаються з імені атрибута і його значення для певної сутності. Атрибути, за якими можна однозначно відрізнити одну сутність від іншої називаються ключовими атрибутами.

Кожна сутність може характеризуватися декількома ключовими атрибутами. Клас взаємозв'язків в IDEF1 являє собою сукупність взаємозв'язків між сутностями. Взаємозв'язок між двома окремими сутностями вважається існуючої в тому випадку, клас атрибутів однієї сутності містить ключові атрибути іншої сутності. Кожен з вищеописаних класів має своє умовне графічне відображення, згідно з методологією IDEF1. Розглянемо IDEF1 інформаційної системи (рис. 3.2).

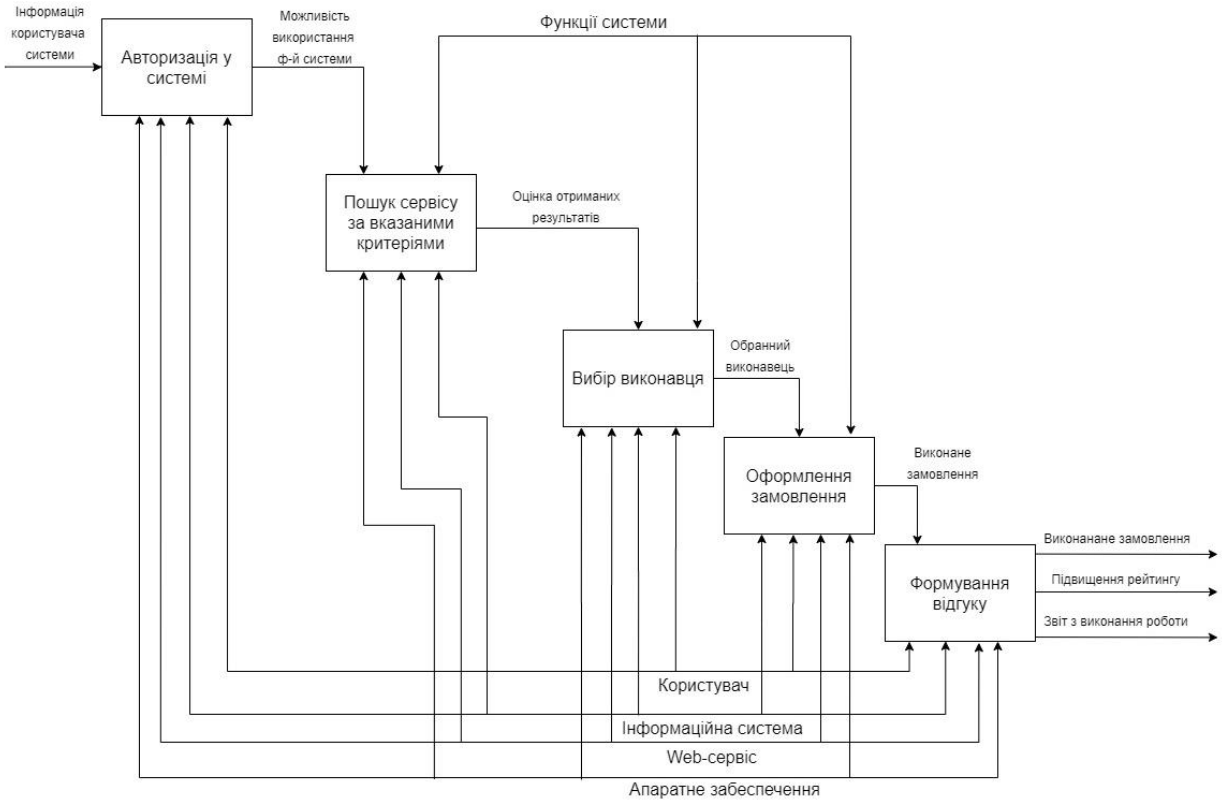

Рисунок 3.2 – IDEF1 проекту

3.2 Моделювання діаграми варіантів використання

Метою діаграми використання є відображення динамічного аспекту системи. Однак це визначення є занадто загальним, щоб описати мету, оскільки інші чотири діаграми (діяльність, послідовність, співпраця та

діаграма стану) також мають те саме призначення. Ми розглянемо якесь конкретне призначення, яке відрізнятиме його від інших чотирьох діаграм.

Діаграми випадків використання використовуються для збору вимог системи, включаючи внутрішні та зовнішні впливи [20]. Ці вимоги в основному є вимогами дизайну. Отже, коли система аналізується для збору її функціональних можливостей, готуються випадки використання та визначаються дійові особи.

Коли початкове завдання виконане, діаграми використання моделюються для подання зовнішнього виду.

Цілі діаграм випадків використання можуть бути такими:

– Використовується для збору вимог системи.
– Використовується для отримання зовнішнього вигляду системи.
– Визначте зовнішні та внутрішні фактори, що впливають на систему.
– Показати взаємодію між вимогами акторів.

Розглянемо детальніше акторів та варіанти використання в табл. 3.1 та табл. 3.2.

Таблиця 3.1 – Опис акторів

№	Назва	Опис
1	Замовник	Користувач, що має можливість створити замовлення в інформаційній системі.
2	Виконавець	Користувач, що має можливість виконувати замовлення, створювати портфоліо та спілкуватися з замовниками.
3	Адміністратор	Користувач, що має найбільші можливості на сайті, а саме редагувати та видаляти дані на сайті.

Таблиця 3.2 – Опис варіантів використання

№	Назва	Опис
1	Авторизація	Модуль дозволяє авторизуватися на сайті.
2	Реєстрація	Модуль дозволяє зареєструватися новому користувачеві на сайті.
3	Редагування інформації на сайті	Дозволяє змінювати загальну інформацію на сайті.
4	Перегляд замовлень та користувачів	Модуль дозволяє переглядати зареєстрованих користувачів та замовлень на сайті.
5	Створення замовлення	Модуль, що дозволяє замовнику створювати нові замовлення.
6	Формування відгуку	Формування відгуку після виконання замовлення.
7	Додавання послуг	Модуль додавання послуг для власного кабінету.
8	Редагування даних на сторінці	Кожен користувач має можливість редагувати особисту інформацію на сайті.

Сформувавши представлення про інформацію та функціонал інформаційної системи було сформовано діаграма варіантів використання (рис. 3.3).

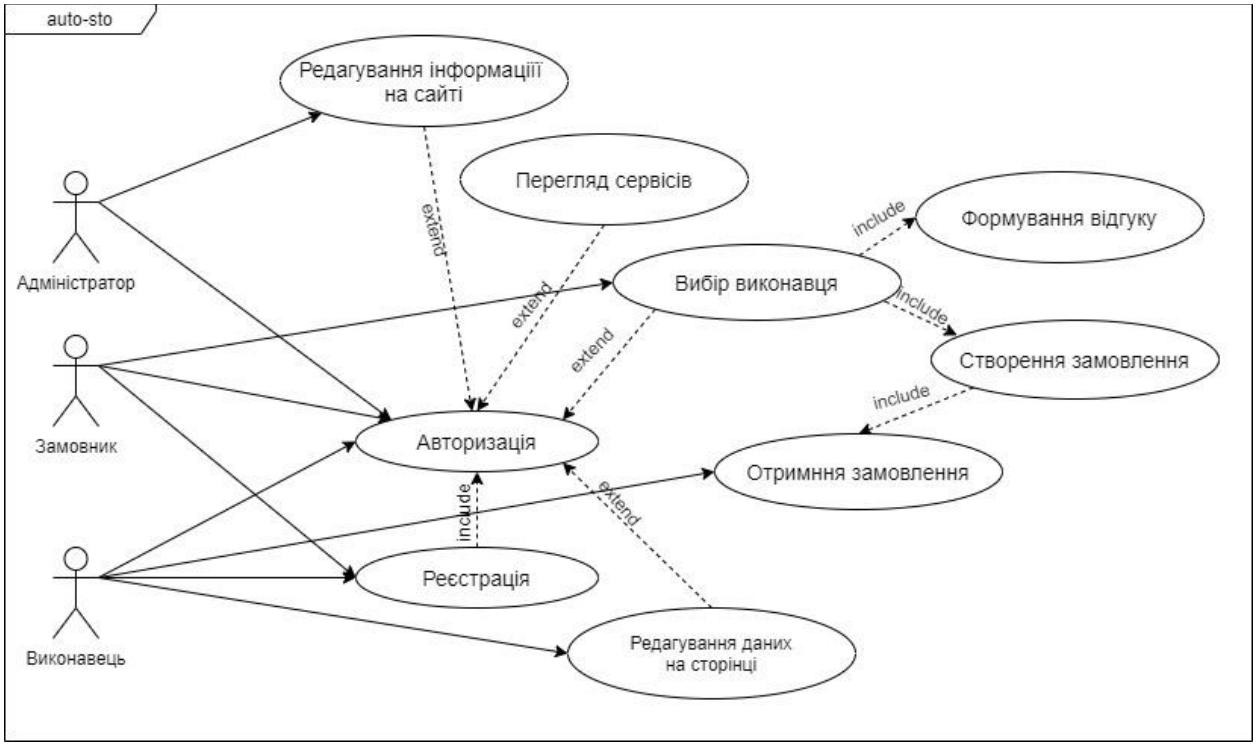

Рисунок 3.3 – Діаграма варіантів використання

3.3. Проектування бази даних

Основні користувачі системи – замовник та фахівець. За допомогою цієї системи вони можуть отримувати різні відомості про виконавців та замовників, послуги та інше.

Зобразимо відносини в ER-діаграмі (рис. 3.4). Проаналізувавши сутність, використовувані в моделі інформаційної системи, перейдемо до реалізації структури БД. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження в табл. 3.3.

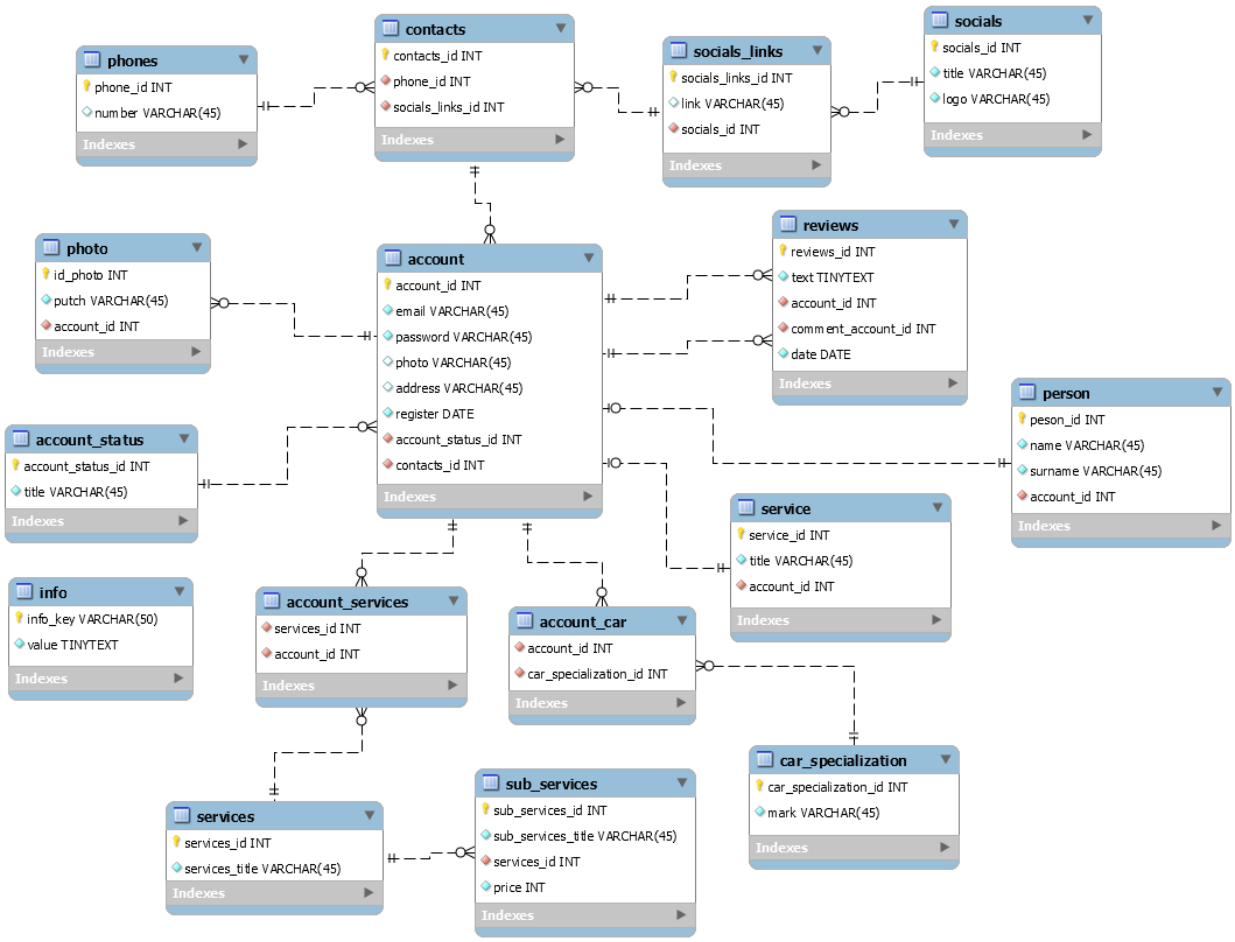

Рисунок 3.4 – ER-діаграма

Таблиця 3.3 – Інформація за таблицями ER-діаграми

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	account	account_id	Ідентифікатор аккаунта	INTEGER	PK	Не пустий
		email	Електронна пошта	VARCHAR(100)		Не пустий
		password	Пароль від аккаунта	VARCHAR(100)		Не пустий
		photo	Фото аккаунта	VARCHAR(100)		
		address	Адреса	VARCHAR(100)		
		register	Дата реєстрації	DATE		Не пустий
		account_status_id	Ідентифікатор статусу аккаунта	INTEGER	FK	Не пустий
		contacts_id	Ідентифікатор контактів	INTEGER	FK	Не пустий
2	account_status	account_status_id	Ідентифікатор статусу аккаунта	INTEGER	PK	Не пустий
		account_status_title	Назва статусу	VARCHAR(100)		Не пустий
3	person	peson_id	Ідентифікатор спеціаліста	INTEGER	PK	Не пустий
		name	Ім’я	VARCHAR(100)		Не пустий
		surname	Прізвище	VARCHAR(100)		Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
4	service	service_id	Ідентифікатор сервісу	INTEGER	PK	Не пустий
		title	Назва сервісу	VARCHAR(100)		Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
5	photo	id_photo	Ідентифікатор зображення	INTEGER	PK	Не пустий
		putch	Посилання на файл	VARCHAR(100)		Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
6	contacts	contacts_id	Ідентифікатор контакту	INTEGER	PK	Не пустий
		phone_id	Ідентифікатор телефону	INTEGER	FK	Не пустий
		socials_links_id	Ідентифікатор посилання соціальної мережі	INTEGER	FK	Не пустий
7	phones	phone_id	Ідентифікатор телефону	INTEGER	PK	Не пустий
		number	Номер телефону	VARCHAR(100)		Не пустий
8	socials_links	socials_links_id	Ідентифікатор посилання соціальної мережі	INTEGER	PK	Не пустий
		link	Посилання на соціальну мережу	VARCHAR(100)		Не пустий
		socials_id	Ідентифікатор соціальної мережі	INTEGER	FK	Не пустий
9	socials	socials_id	Ідентифікатор соціальної мережі	INTEGER	PK	Не пустий
		title	Назва соціальної мережі	VARCHAR(100)		Не пустий
		logo	Логотип соціальної мережі	VARCHAR(100)		Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
10	account_services	services_id	Ідентифікатор спеціалізації	INTEGER	FK	Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
11	services	services_id	Ідентифікатор спеціалізації	INTEGER	PK	Не пустий
		services_title	Назва спеціалізації	VARCHAR(100)		Не пустий
12	sub_s ervices	sub_services_id	Ідентифікатор послуги	INTEGER	PK	Не пустий
		sub_services_title	Назва послуги	VARCHAR(100)		Не пустий
		services_id	Ідентифікатор спеціалізації	INTEGER	FK	Не пустий
		price	Ціна спеціалізації	INTEGER		Не пустий
13	account_car	account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		car_specialization_id	Ідентифікатор марки авто	INTEGER	FK	Не пустий
14	car_specialization	car_specialization_id	Ідентифікатор марки авто	INTEGER	PK	Не пустий
		mark	Марка авто	VARCHAR(100)		Не пустий
15	reviews	reviews_id	Ідентифікатор відгуку	INTEGER	PK	Не пустий
		text	Текст відгуку	TEXT		Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		comment_account_id	Ідентифікатор аккаунта що залишив відгук	INTEGER	FK	Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
16	info	info_key	Ключ запису	VARCHAR(100)	PK	Не пустий
		value	Значення	TEXT		Не пустий

3.4. Моделювання діаграм діяльності

Діаграма діяльності – це технологія, що дозволяє описувати логіку процедур, бізнес-процеси і потоки робіт. У багатьох випадках вони нагадують блок-схеми, але принципова різниця між діаграмами діяльності і нотацією блок-схем полягає в тому, що перші підтримують паралельне процеси.

Діаграма діяльності дозволяє будь-кому, хто виконує цей процес, вибирати порядок дій. Іншими словами, діаграма тільки встановлює правила обов'язкової послідовності дій, які я повинен виконувати [21]. Це важливо для моделювання бізнес-процесів, оскільки ці процеси часто виконуються паралельно. Такі діаграми також корисні при розробці паралельних алгоритмів, в яких незалежні потоки можуть виконувати роботу паралельно.

На рисунку 3.5-9 зображено діаграми діяльності модулів інформаційної системи фріланс-біржі.

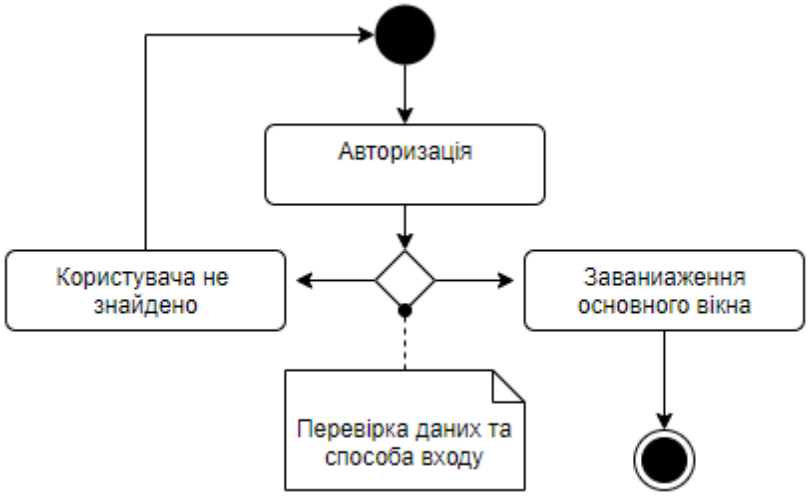

Рисунок 3.5 – Діаграма діяльності модулю авторизації

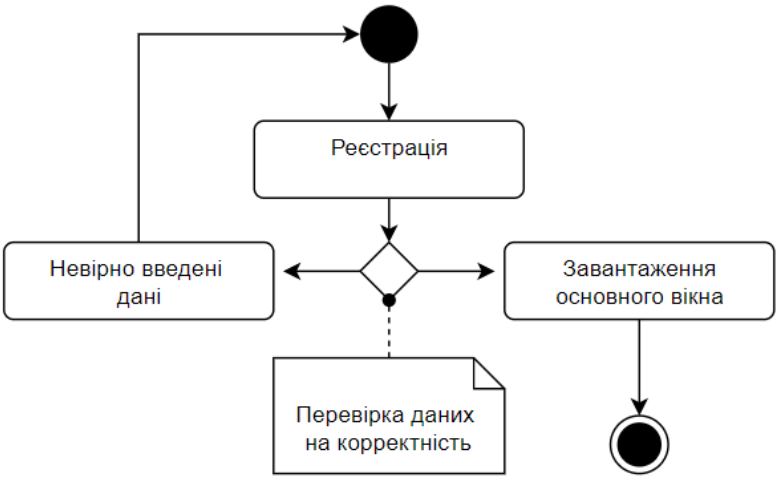

Рисунок 3.6 – Діаграма діяльності модулю реєстрації

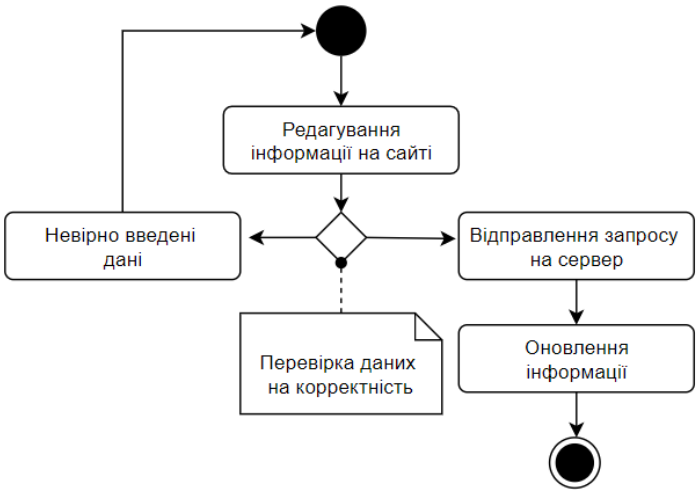

Рисунок 3.7 – Діаграма діяльності модулю редагування інформації на сайті

Рисунок 3.8 – Діаграма діяльності модулю перегляду інформації

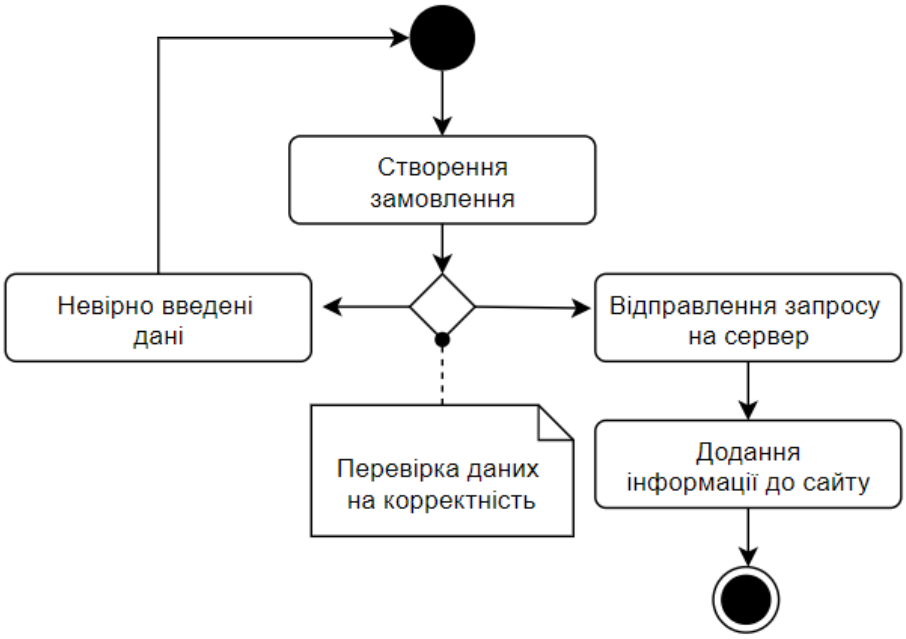

Рисунок 3.9 – Діаграма діяльності модулю створення замовлення

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

4.1 Програмна реалізація

Програмна реалізація додатку була виконана за допомогою фреймворків Laravel та Vue.js [22]. Розглянемо детальніше структуру інформаційної системи (табл. 4.1).

Таблиця 4.1 – Таблиця рівнів доступу

№	Пакет	Короткий опис
1	«app»	Містить код ядра додатку.
2	«bootstrap»	Містить файли, які завантажують фреймворк і налаштовують автозагрузку.
3	«config»	Містить всі конфігураційні файли додатку
4	«database»	Містить міграції і класи для наповнення початковими даними БД.
5	«public»	Містить файл index.php, який є вхідною точкою для всіх запитів, що надходять в додаток. Також ця папка містить ресурси, такі як зображення, JavaScript, CSS.
6	«resources»	Містить початковий код, а також сирі, некомпільовані ресурси, такі як LESS, SASS, JavaScript.
7	«routes»	Містить всі визначення маршрутів додатку.
8	«storage»	Містить скомпільовані Blade-шаблони, файл-сесії, кешу файлів і інші файли, створені фреймворком.
9	«tests»	Містить автотести.

Інформаційна система надання послуг з ремонту автомобілей має певну кількість типів користувачів, а саме такі як адміністратор, «СТО/працівник» та «Звичайний користувач».

Розглянемо розподіл функціоналу детальніше в табл. 4.2.

Таблиця 4.2 – Функціонал за групами користувачів

№	Користувач	Можливості користувача
1	Адміністратор	Перегляд загальної інформації.
		Підтримка сайту.
		Блокування користувачів.
2	СТО/працівник	Створення власної інформаційної сторінки.
		Редагування даних.
		Формування розділу «Навички».
		Листування з користувачами системи.
		Написання повідомлення адміністратору.
3	Звичайний користувач	Створення власної інформаційної сторінки;
		Редагування даних;
		Листування з користувачами системи;
		Використання вбудованого калькулятору для обрахування ціни ремонту автомобіля;
		Оформлення замовлення послуги або консультації;
		Залишити відгук виконавцю за виконану роботу;
		Написання повідомлення адміністратору.

Для детального представлення додатку виконаємо детальний опис кожної сторінки подальшого для прототипування.

4.1.1. Головна сторінка

Сторінка з основною довідкою та можливостями інформаційної системи надання послуг з ремонту автомобілів. Дана сторінка є першою на яку потрапляє користувач.

Кількість автомобілістів збільшується з кожним роком, зростає попит на ремонт в наслідок чого відкриваються нові СТО. Користувач буде проінформований про останні новини та ознайомлений з переліком головних переваг даного сервісу.

4.1.2. Реєстрація

Сторінка для реєстрації нових користувачів. Користувач повинен ввести певний перелік даних для створення акаунту. Якщо аккаунт створюється для СТО чи команди з ремонту автомобілей, то для реєстрації потрібно заповнити такі поля:

- Назва сервісу;
- Електронна пошта;
- Пароль;

Якщо аккаунт відноситься до одного користувача, то список полів видозмінюється та розширюється.

Перелік даних необхідних для реєстрації:

- Ім’я;
- Прізвище;
- Електронна пошта;
- Пароль;
- Дата народження;
- Стать;
- Напрямок роботи.

4.1.3. Авторизація

Сторінка для авторизації. В незалежності від типу сторінки, користувач повинен ввести такі дані:

- Електронна пошта;
- Пароль.

4.1.4. Особистий кабінет

Сторінка з інформацією про користувача. На даній сторінці є можливість перегляду основних розділів, а саме контактної інформації, стажу роботи, напрямок роботи фахівця чи СТО та ціни за перелік послуг.

4.1.5. Налаштування акаунту

Сторінка для редагування основної інформації про користувача та додаткове її налаштування.

4.1.6. Вибір майстерні

Дана сторінка потрібна для підбору СТО чи фахівця для ремонту автомобілю. Можна виконувати пошук та фільтрацію даних за певними параметрами.

Параметри пошуку:

- Тип роботи
- Діапазон ціни
- Місце роботи

4.1.7. Онлайн калькулятор

Онлайн калькулятор – додатковий функціонал для обрахування комплексу роботи по ремонту автомобіля. На сторінці можна ввести виконавця роботи, а потім перелік потрібної роботи. Ціна буде вираховувати в залежності від вказаної ціни самого фахівця на власній сторінці, в залежності від типу завдання та складності.

4.1.8. Листування

За допомогою листів можна вирішувати багато важливих питань. Даний функціонал допоможе користувачам розвивати партнерські відносини, інформувати про результати і завдання, ділитися новинами та інше.

4.2 Використання програмного додатку

4.2.1. Користувач – спільні сторінки та власник авто

Розглянемо головну сторінку інформаційної системи підбору сервісів на рисунку 4.1 – 3.

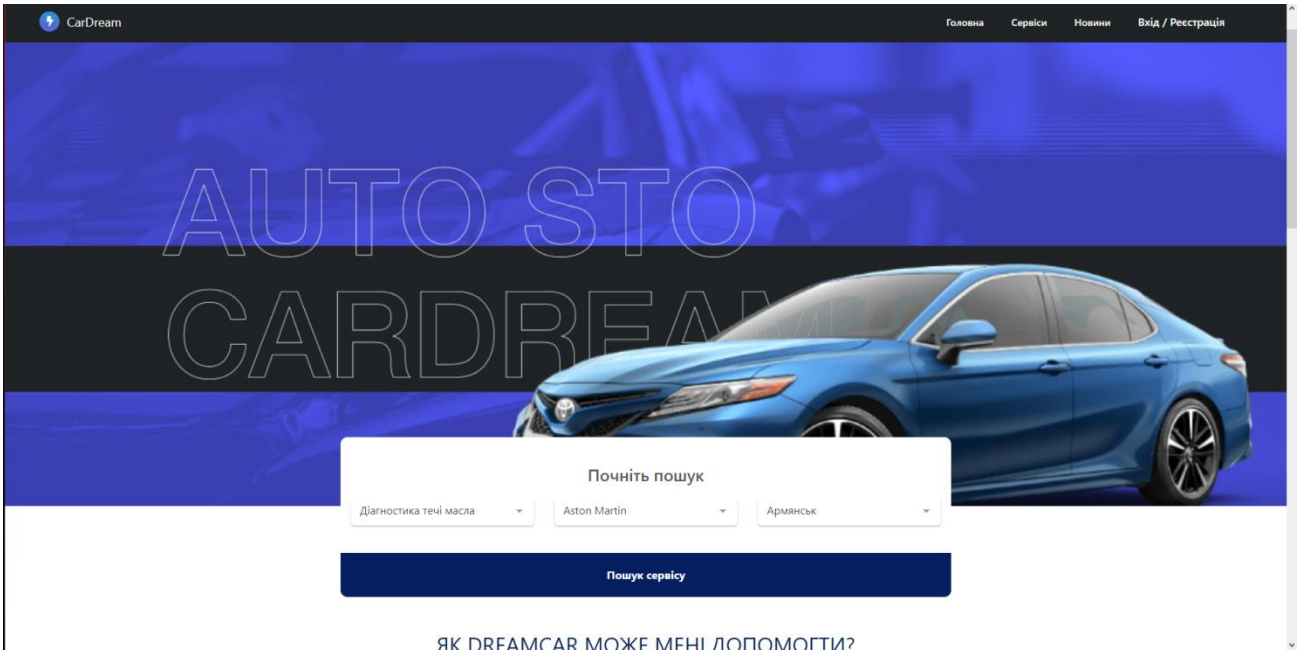

Рисунок 4.1 – Головна сторінка сервісу

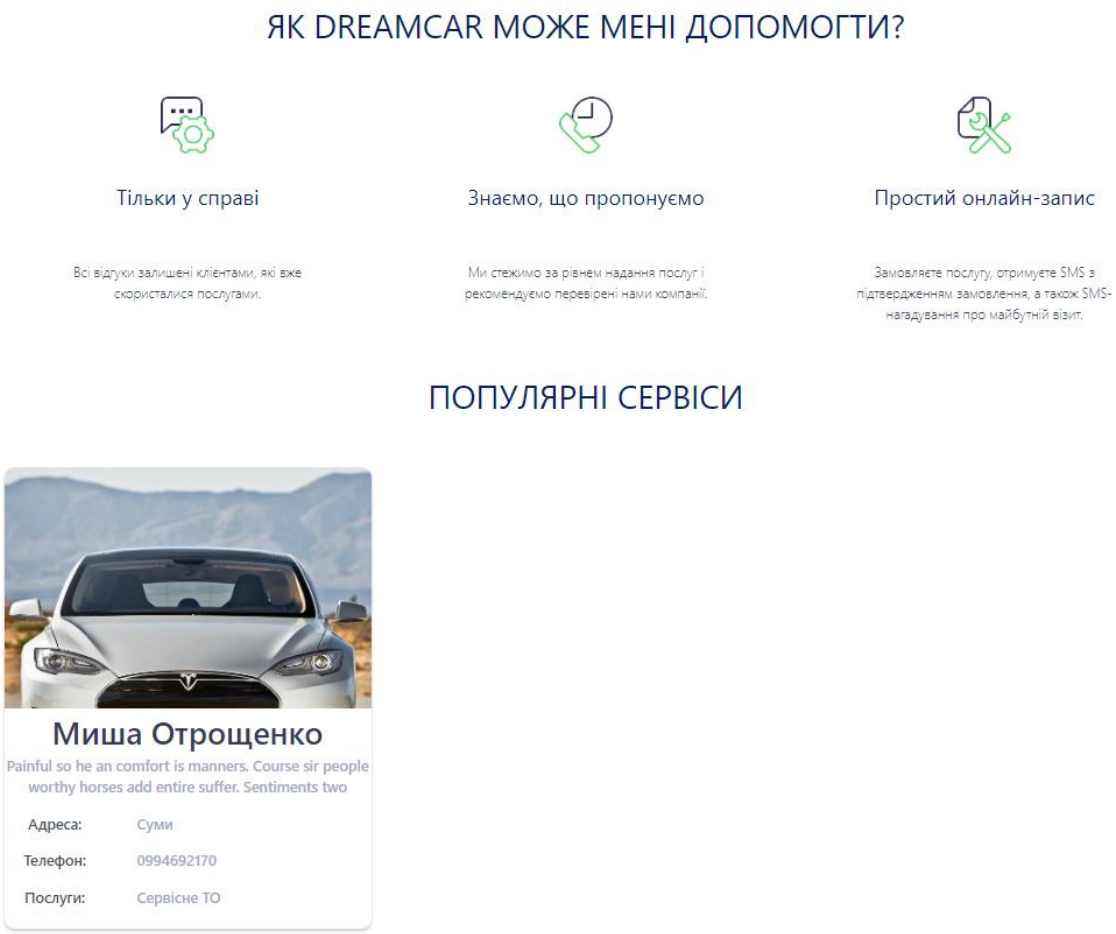

Рисунок 4.2 – Загальна інформація про інформаційну систему

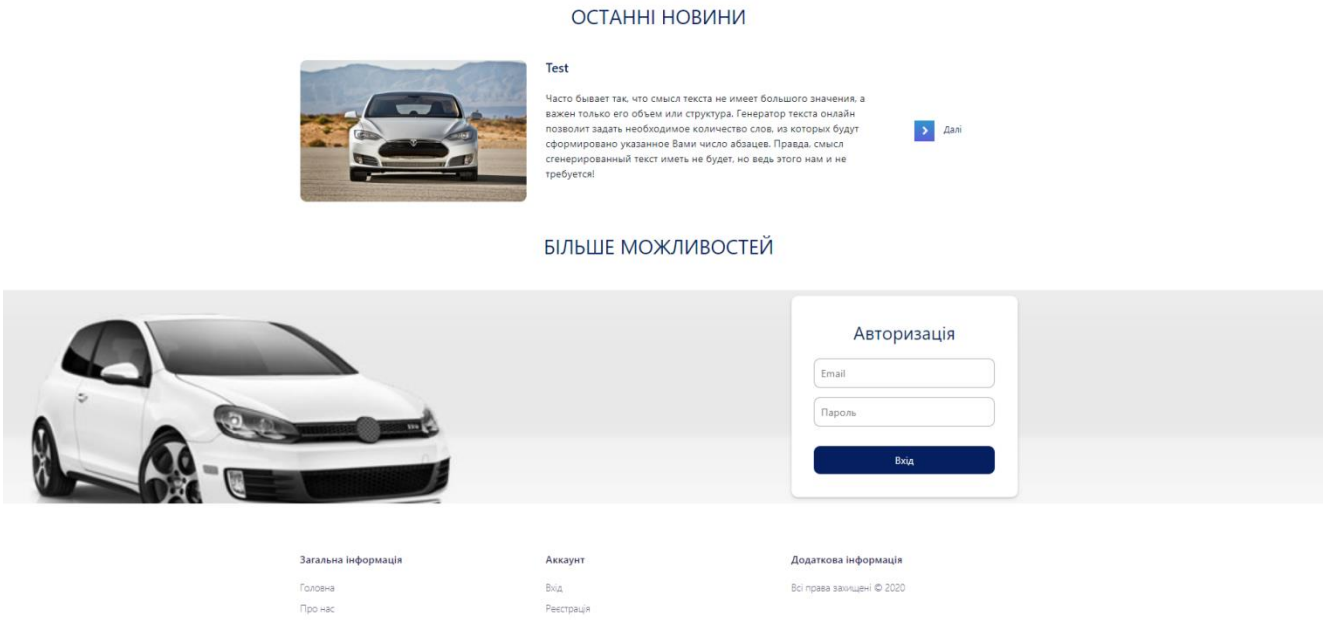

Рисунок 4.3 – Блок з новинами та авторизацією

Важливим пунктом є функція пошуку за типом роботи, маркою машини та містом (рис. 4.4).

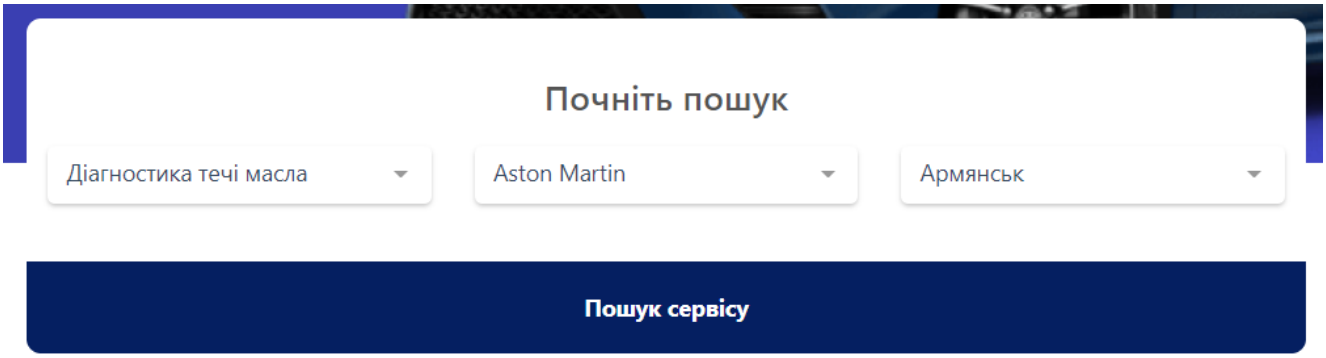

Рисунок 4.4 – Функція пошуку на головній сторінці

Розглянемо реєстрацію. Є можливість зареєструватися як автовласник, майстер та сервісний центр (рис. 4.5-6).

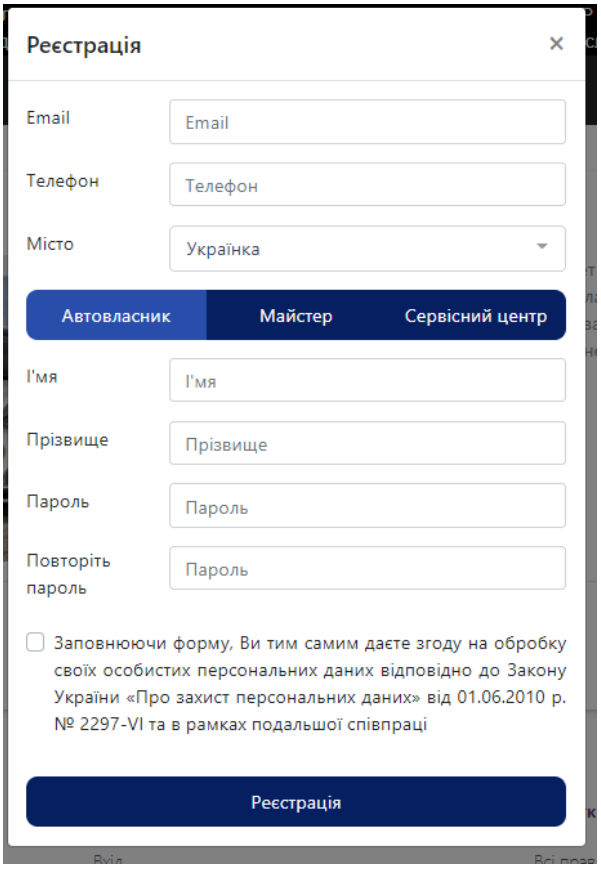

Рисунок 4.5 – Реєстрація як автовласник

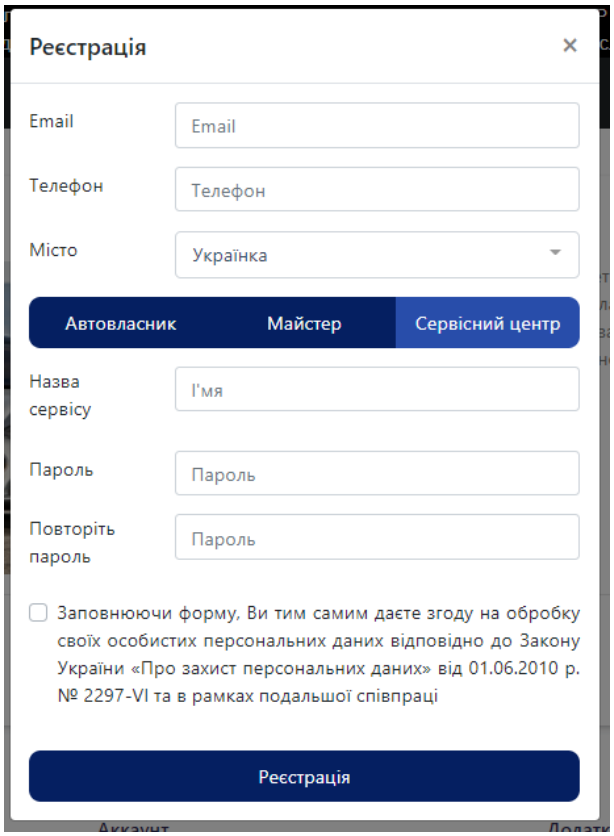

Рисунок 4.6 – Реєстрація як сервісний центр

Після реєстрації, користувач потрапляє на сторінку кабінету, а саме «Загальна інформація» (рис. 4.7).

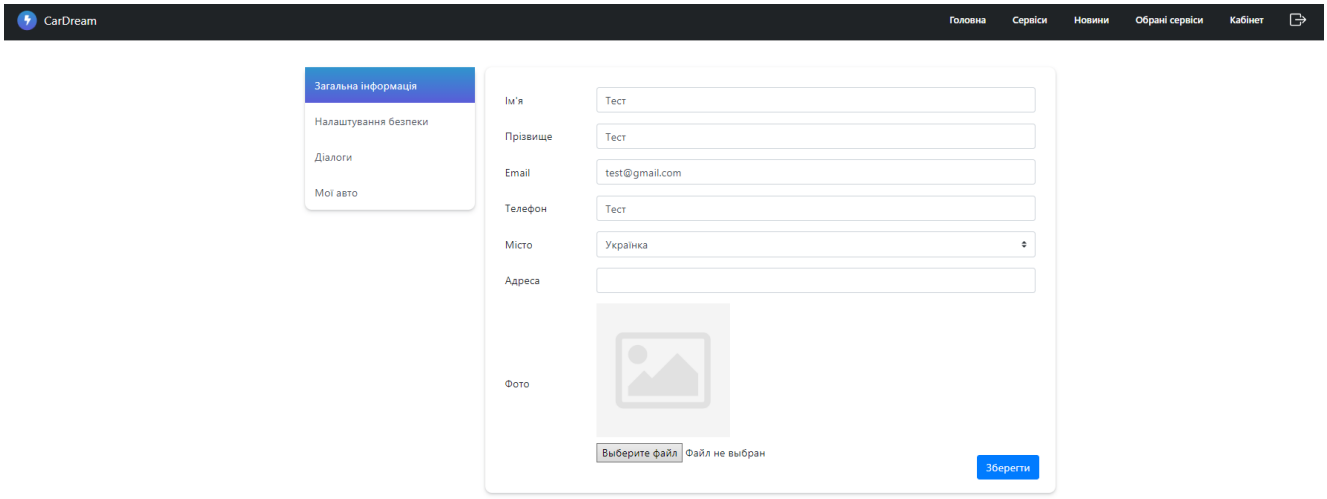

Рисунок 4.7 – Сторінка «Кабінет», пункт «Загальна інформація»

На вкладці «Мої авто», можна додати автомобіль та загальну інформацію (рис. 4.8-9).

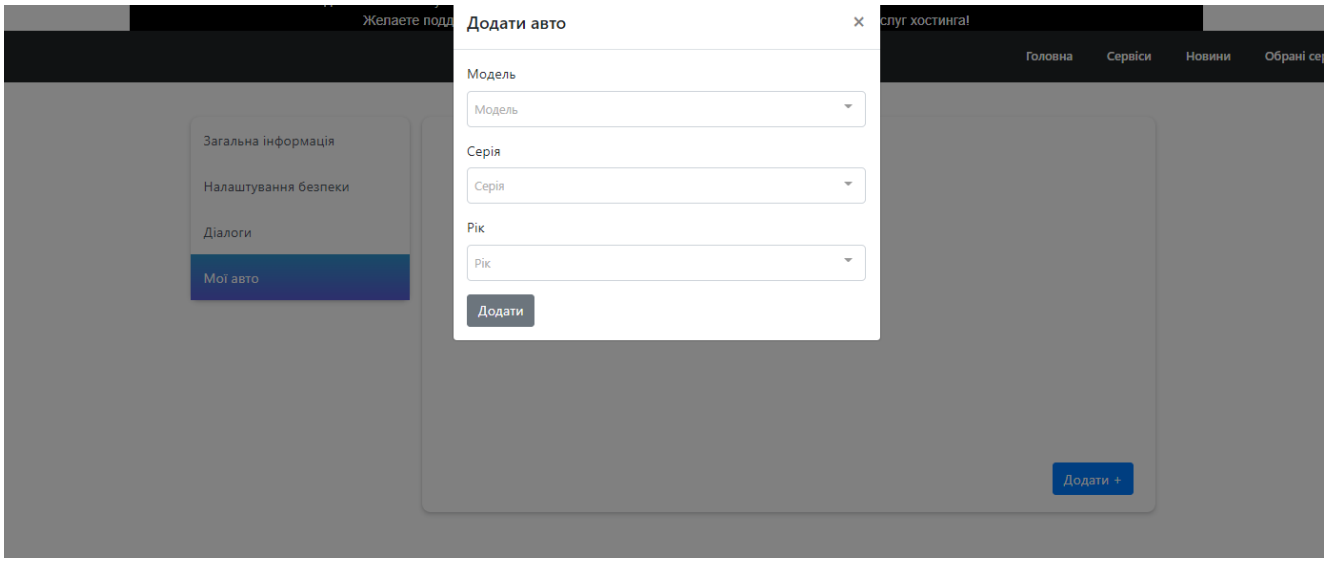

Рисунок 4.8 – Сторінка «Мої авто», додавання авто

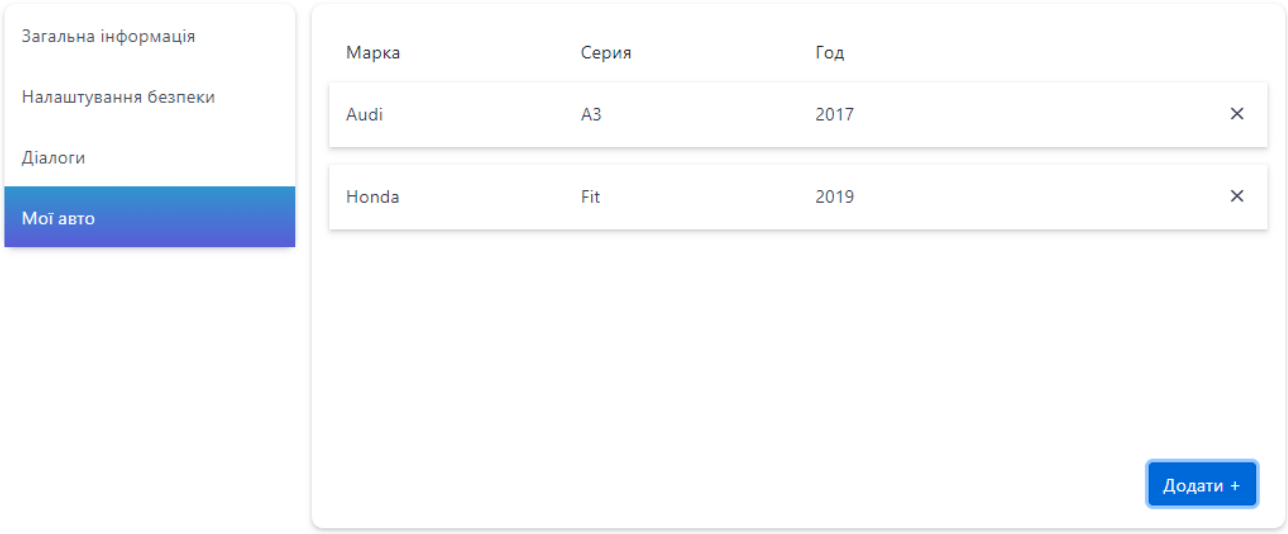

Рисунок 4.9 – Додані автомобілі

На сторінці «Сервіси» можна виконувати пошук за такими пунктами:

– Послуга;

– Авто;

– Місто;

– Назва сервісу;

Також йде фільтрація за типом акаунту – сервісний центр чи майстер (рис. 4.10).

Рисунок 4.10 – Сторінка пошуку сервісу

Сторінка сервісу представлена на рисунку 4.11.

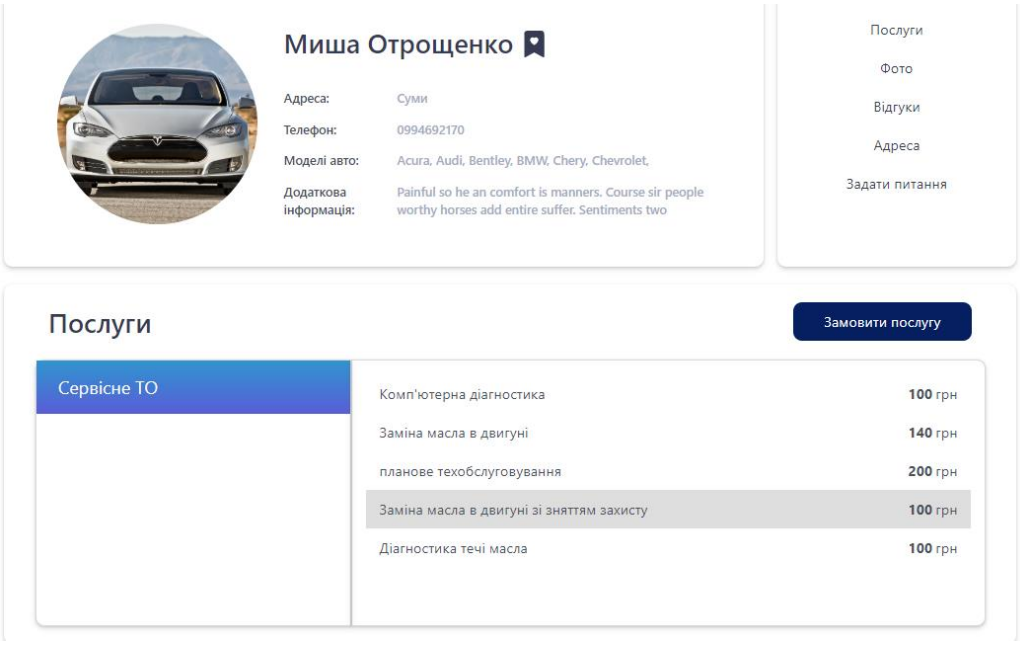

Рисунок 4.11 – Сторінка сервісу

На сторінці сервісу можна оформити замовлення з інформацією обраною на сторінці іншого користувача (сервісний центр/майстер) (рис. 4.12).

Замовлення послугу йде за такими пунктами:

– Модель;
– Серія;
– Рік;
– Контактна інформація;
– Коментар.

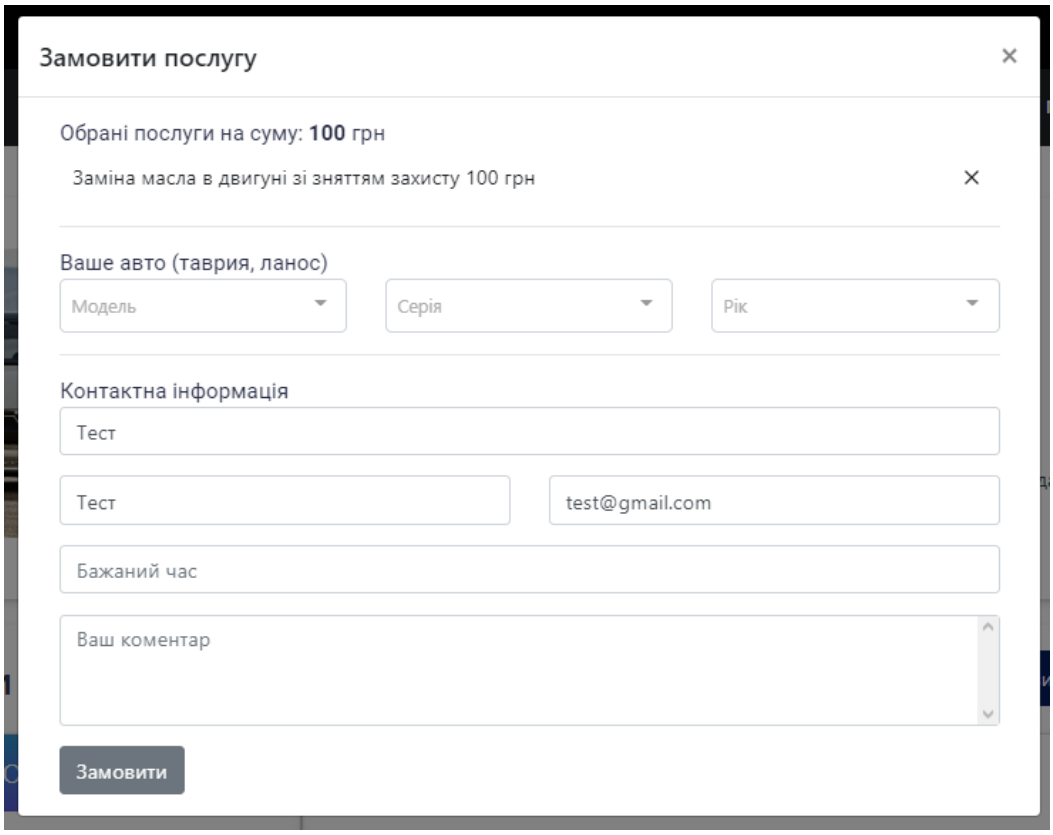

Рисунок 4.12 – Замовлення послуг

На головній сторінці можна продивитися:

– Послуги;
– Фото;
– Адреса (рис. 4.13);

– Відгуки (рис. 4.14);

– Задати запитання (рис. 4.15-16);

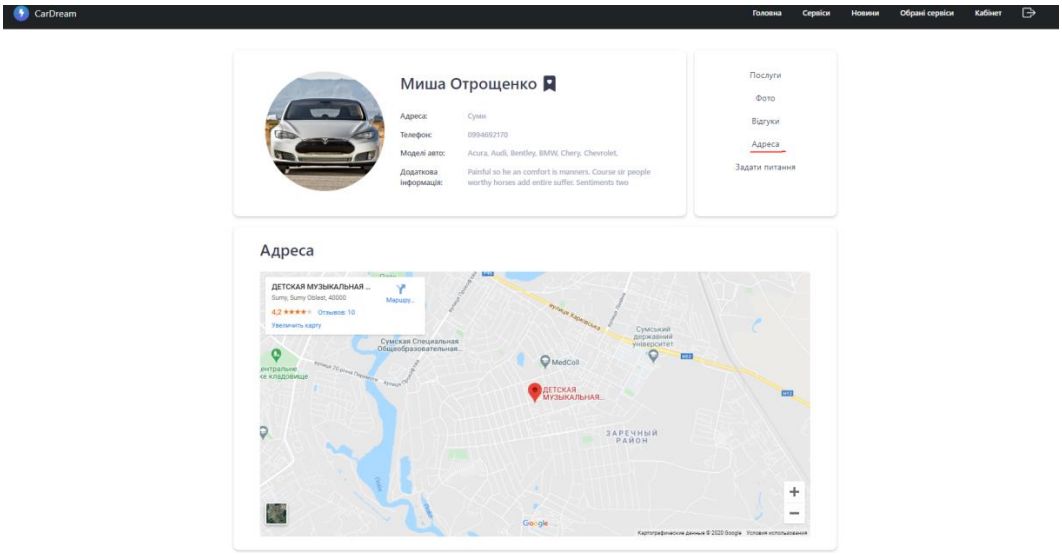

Рисунок 4.13 – Відображення адреси

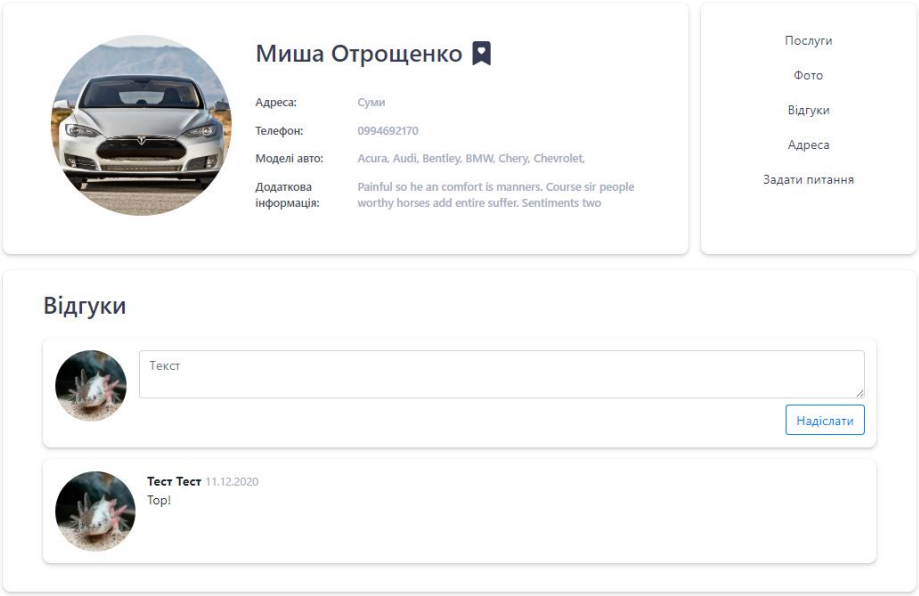

Рисунок 4.14 – Коментарі на сторінці

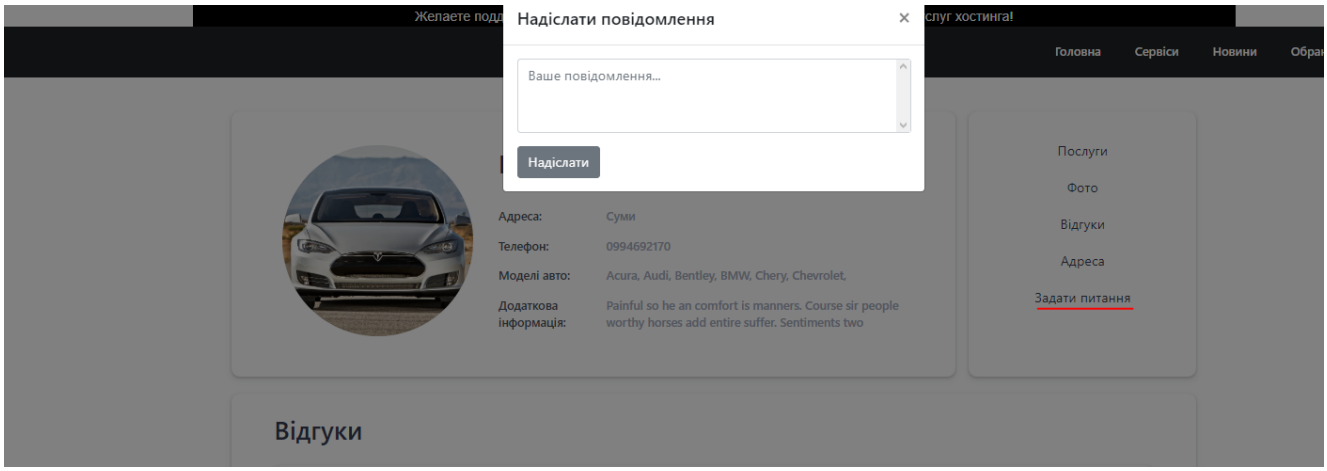

Рисунок 4.15 – Повідомлення

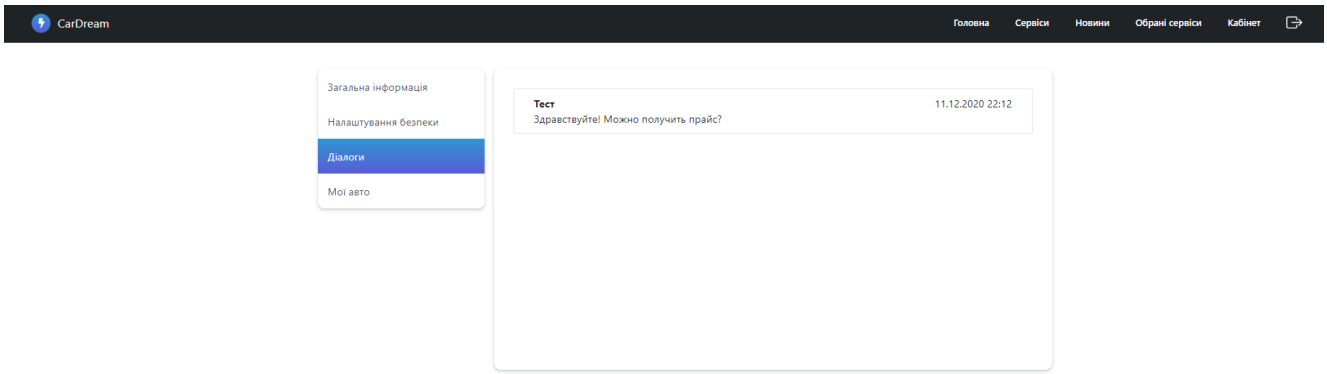

Рисунок 4.16 – Відображення діалогів

Сторінку сервісу чи майстра можна додати до бажаних, що продемонстровано на рисунку 4.17-18.

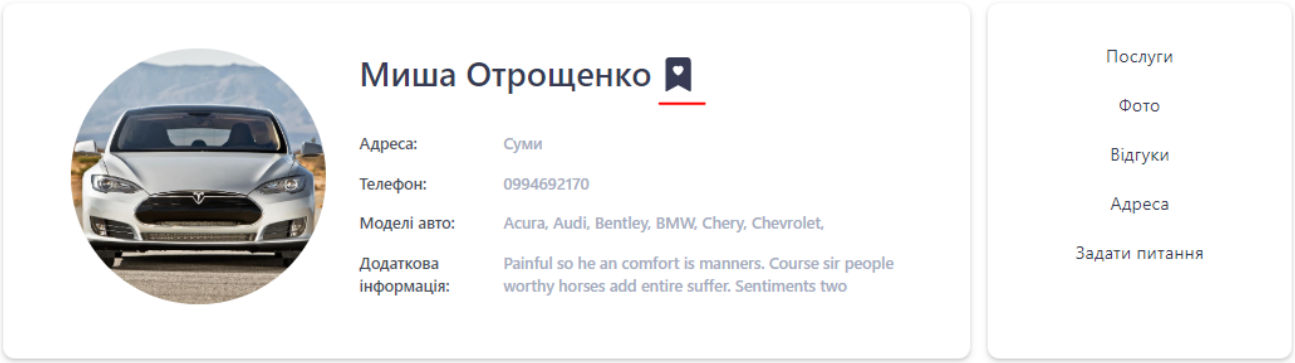

Рисунок 4.17 – Додавання до бажаних

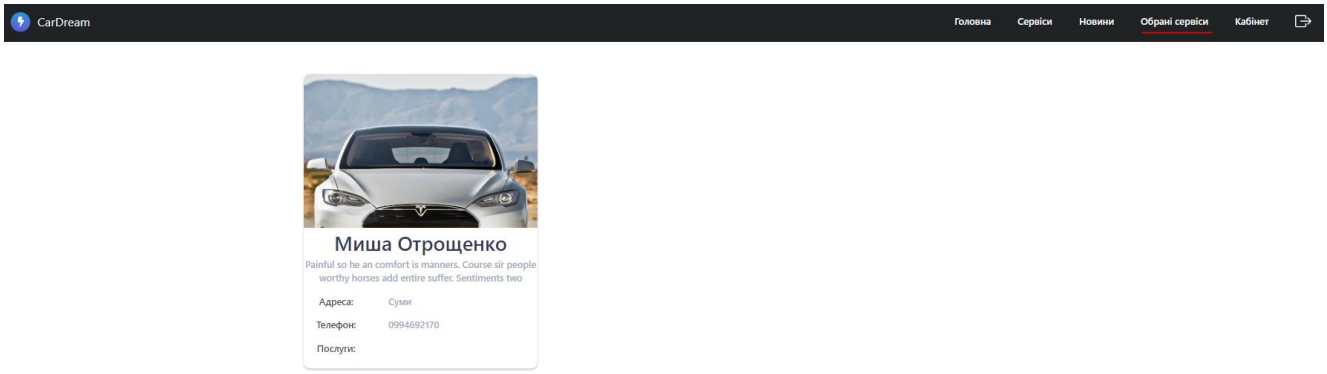

Рисунок 4.18 – Відображення аккаунту в бажаних

4.2.2. Користувач – сервіс чи механік

Відображення можливостей користувача типу «механік» чи «сервіс», продемонстровано на рисунках 4.19-21.

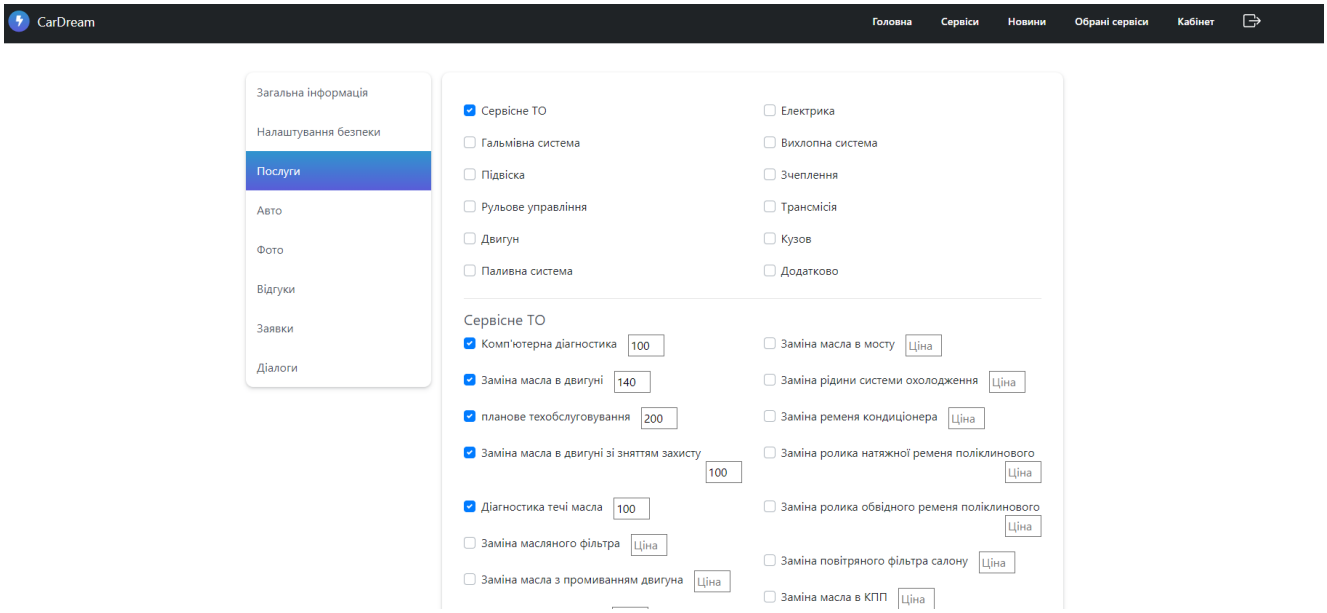

Рисунок 4.19 – Пункт «Послуги»

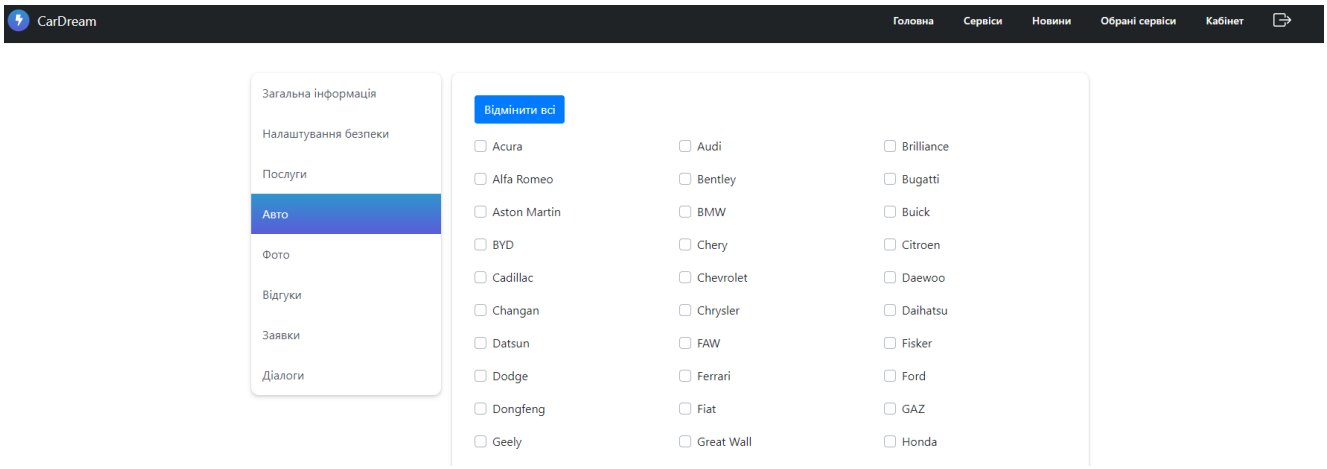

Рисунок 4.20 – Пункт «Авто»

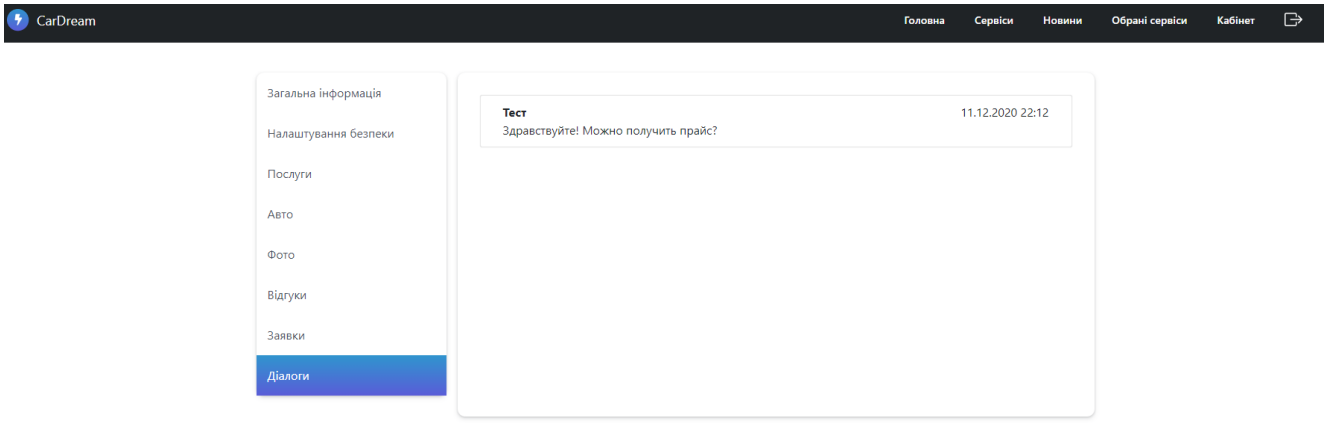

Рисунок 4.21 – Пункт «Діалог»

4.2.3 Адміністратор

Для переходу до адміністративної панелі (рис. 4.22), потрібно перейти на сторінку авторизацію.

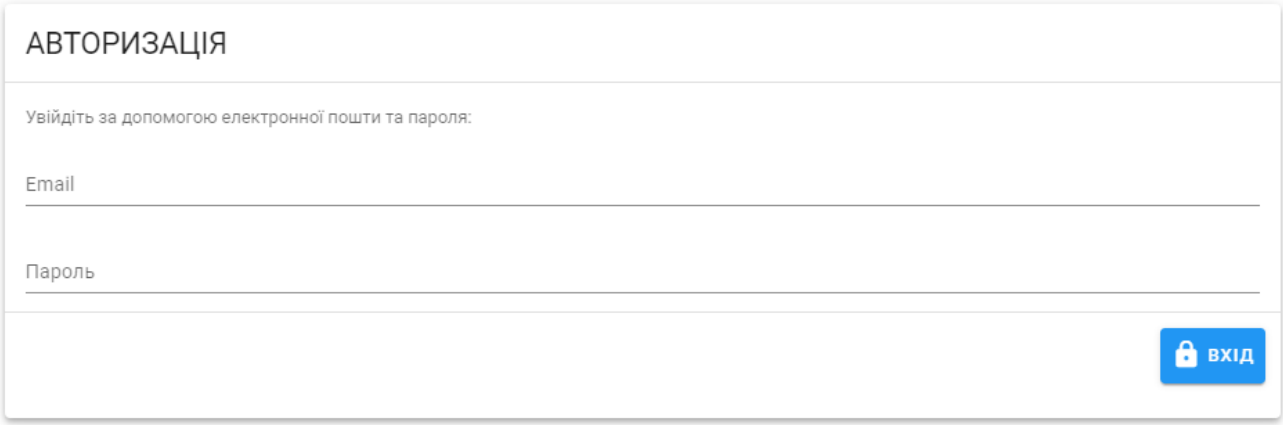

Рисунок 4.22 – Сторінка авторизації для адміністратора

Адміністратор доступні такі функції як (рис. 4.23-26):

– Зміни статусу користувачам;
– Редагування послуг та вмінь;
– Додавання новин;
– Перевірка замовлень.

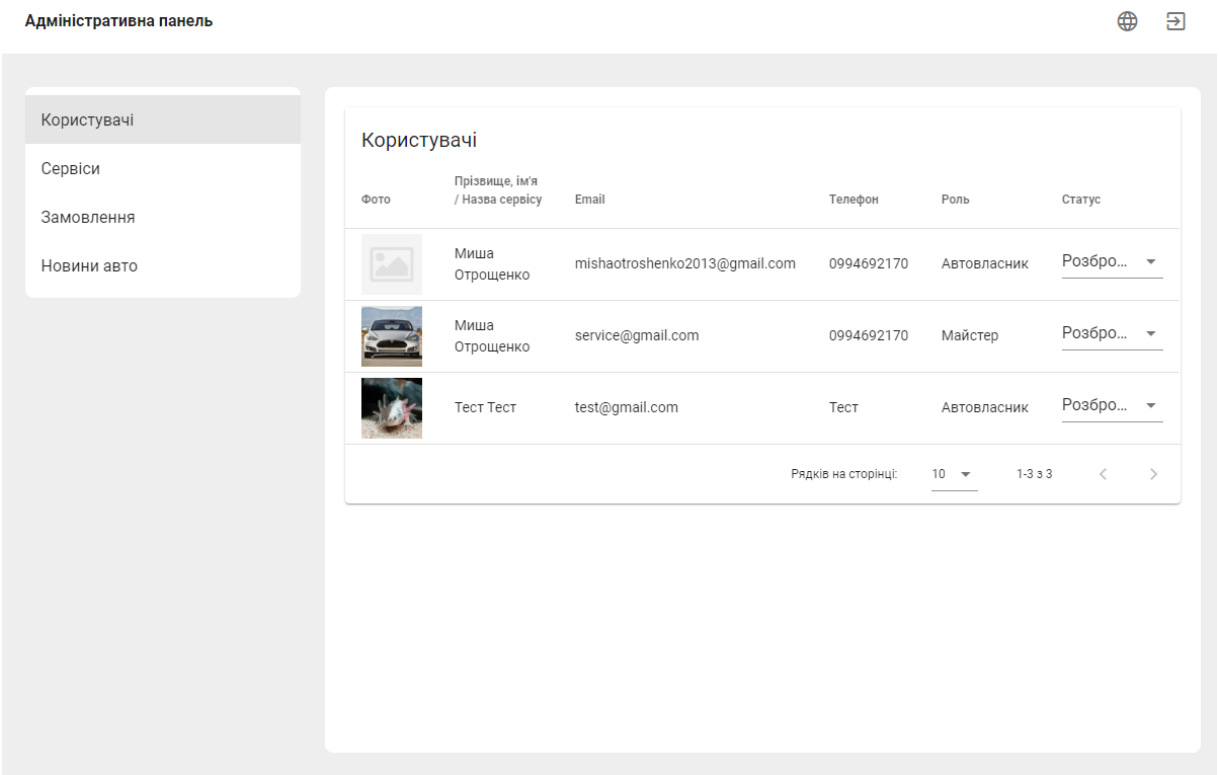

Рисунок 4.23 – Редагування користувачів

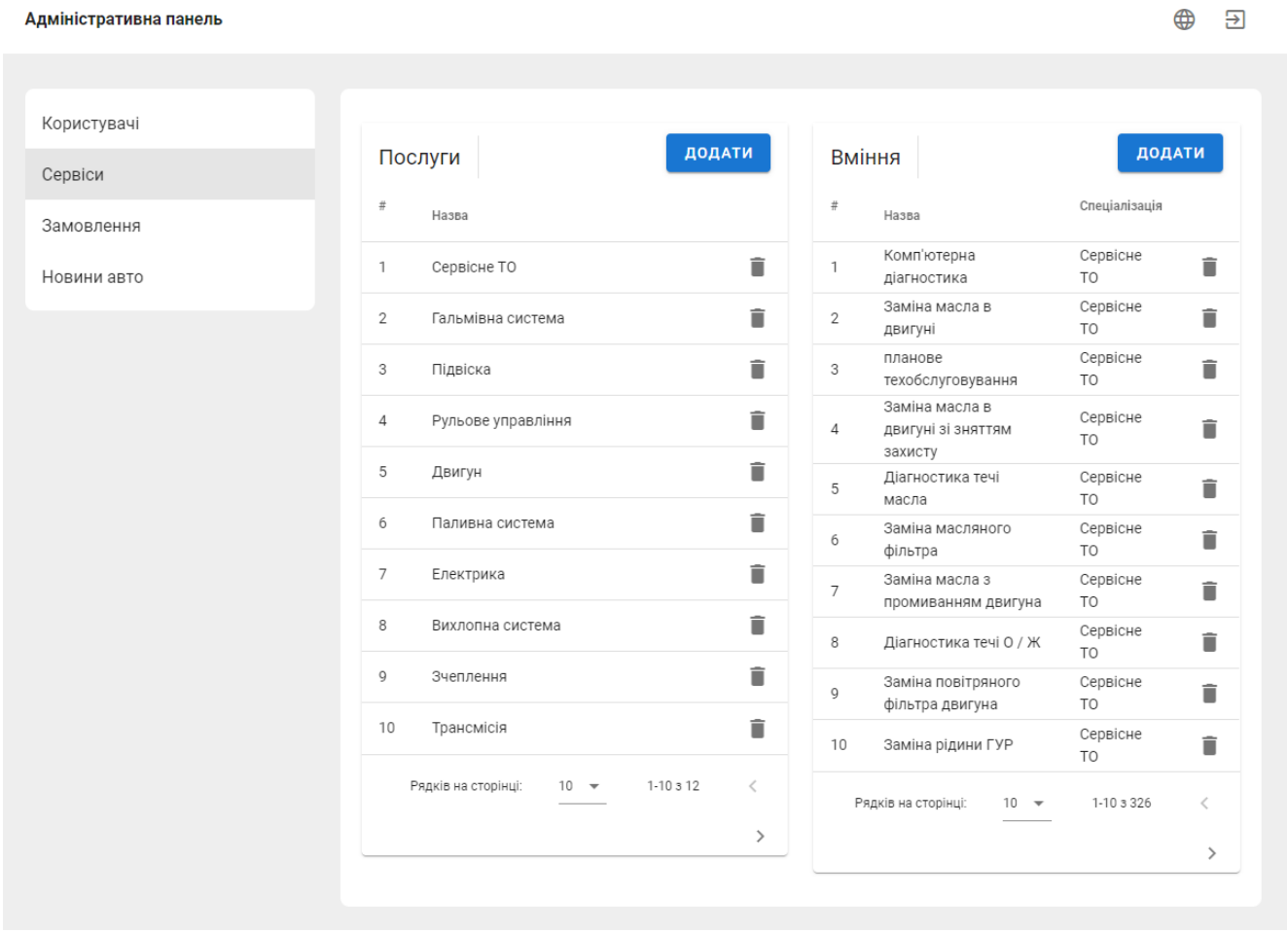

Рисунок 4.24 – Редагування сервісів

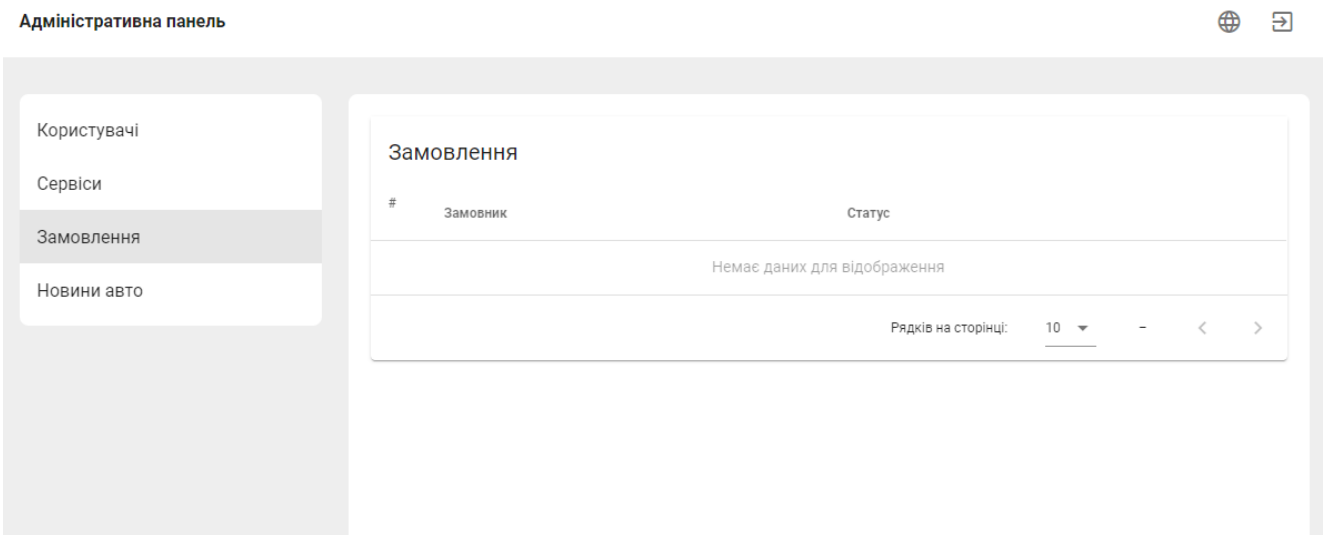

Рисунок 4.25 – Редагування замовлень

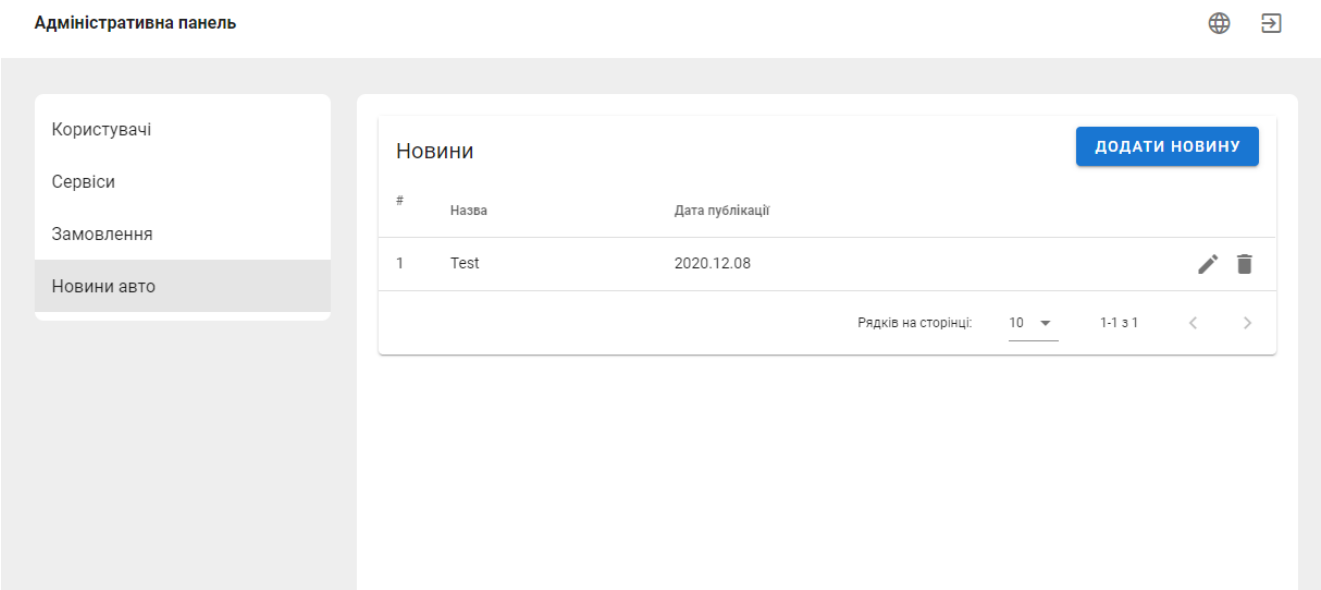

Рисунок 4.26 – Роботи із новинами

ВИСНОВКИ

Під час виконання дипломної роботи та розробки інформаційної системи було проведено дослідження ситуації з автосервісами в Україні на сьогоднішній день, а також аналіз відповідних сайтів аналогів з виявленням їх переваг і недоліків.

Грунтуючись на виявлених потребах, було виконано:

- аналіз обраної предметної області;
- огляд останніх актуальних досліджень та сервісів;
- обрані методи для реалізації інформаційної системи;
- проектування системи надання послуг з ремонту автомобілів;
- проектування бази даних;
- реалізацію та детальний опис використання інформаційної системи зі сторони власника автомобіля, механіка та автомобільного сервісу.

Програмна реалізація системи була виконана за допомогою мов програмування JavaScript та PHP, фреймворків Laravel та Vue.js.

В результаті проведених дій було розроблено інформаційну систему, яка є повністю функціональною, а також має інтуїтивно зрозумілий інтерфейс. Виконаний веб-сервіс дає можливість виконувати пошук СТО з можливістю фільтрації по моделі авто та виду поломки, автоматично підраховувати ціну ремонту, переглядати повну інформацію про сервіс, можливість реєстрації та авторизації користувачів через електронну пошту, залишати відгуки, записуватись на ремонт, підтримувати графічні вставки та анімації.

Таким чином створена система відповідає всім вимогам технічного завдання.

СПИСОК ЛІТЕРАТУРИ

1. Переваги та недоліки сертифікованих СТО – https://shop.febest.ua/ru/blog/preimuschestva-i-nedostatki-sertificirovanyh-sto-pjat-praktichnyh-sovetov-ot-febest-kak-vybrat-luchshee.html.

2. Вибираємо СТО – https://www.032.ua/list/58471.

3. У чому перевага спеціалізованої СТО – https://innet.org.ua/article-229-v-chem-preimuschestvo-spetsializirovannoy-sto.html.

4. Як вибрати СТО: 5 основних правил – https://www.autocentre.ua/opyt/avtoservis/kak-vybrat-sto-5-osnovnyh-pravil-367236.html.

5. Станція техобслуговування - види, плюси і мінуси – https://zamena-masla-spot.ru/useful/stanciya-tehobsluzhivaniya-vidy-plyusy-i-minusy.

6. Як вибрати СТО? Критерії та поради пошуку автосервісу –https://your-car.com.ua/Novosti/view/164.

7. Як вибрати СТО – https://www.auto-klinika.kh.ua/kak-vybrat-sto.

8. Як вибрати надійне СТО – http://avtosreda.ru/info/auto-insurance/kak-vybrat-nadjezhnoje-sto/.

9. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0). Draft Federal Information Processing Standards Publication 183 ,1993 December 21.

10. INTEGRATION DEFINITION FOR INFORMATION MODELING (IDEF1X), Draft Federal Information Processing Standards Publication 184 1993 December 21.

11. Sheer, A. ARIS-Business Process Modelling / Sheer A. – Springer-Verlag, Berlin,1998.

12. Booch, G. The Unified Modeling Language User Guide / Booch, G., Rumbaugh, J., Jacobson, I. – Second Edition, AddisonWesley, 2005.

13. Методичні вказівки до лабораторних робіт з курсу «Організація баз даних та баз знань». Укладач В.О. Нелюбов. – Ужгород: Видавничий центр ЗакДУ, 2010.

14. Система управління базами даних Access. Навчальний посібник «Організація баз даних і баз знань»/ Укладач В.О. Нелюбов. – Ужгород: Редакційно-видавничій відділ, 2015.

15. Круг Стів. Вебдизайн: Книга Стіва Кола або не змушуй мене думати [Текст] / Стів Круг. — Символ-Плюс, 2001.

16. Райс Ерік. Інформаційно архітектурний підхід до створення успішних веб-сайтів. [Текст] / Ерік Райс. — Addison Wesley, 2000.

17. Xiao, H. Web-based Automobile Sales Management Systemn15.

18. Kim, W. A design and implementation of the real time 4-Channel image processing system for vehicle using the smart phone. Advanced Science Letters. 2017. Vol. 23, No. 4.

19. Shahlol, A., Alix, A., Lagman, A. Web-based automobile service management system for MAS motors LLC: 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018, 19.

20. Cai, Z., Zhang, J. Commodity Trading Platform Based on Big Data Analysis: Advances in Intelligent Systems and Computing, 21.

21. Eller, M., Dave, A., Johnson, C., та ін. Accuracy of 4-Dimensional Computed Tomography for Localization in Primary Hyperparathyroidism. Journal of Surgical Research. 2021. Vol. 257.

22. Behrens, M., Boyle, S., Fingeret, A. L. Evaluation for Primary Hyperparathyroidism in Patients Who Present With Nephrolithiasis. Journal of Surgical Research. 2021. Vol. 257.

ДОДАТОК А

2014_10_12_000000_create_users_table.php

Міграція створення таблиці аккаута.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name')->nullable();
            $table->string('surname')->nullable();
            $table->string('phone');
            $table->string('address')->nullable();
            $table->string('photo')->default('/img/no-image.png');
            $table->string('service_name')->nullable();
            $table->foreignId('city_id');
            $table->foreignId('user_role_id');
            $table->string('email')->unique();
            $table->text('description')->nullable();
            $table->string('password');
            $table->timestamps();
        });

        Schema::table('users', function (Blueprint $table) {
            $table->index('city_id');
            $table->foreign('city_id')->references('id')->on('city');
        });

        Schema::table('users', function (Blueprint $table) {
            $table->index('user_role_id');
            $table->foreign('user_role_id')->references('id')->on('user_roles');
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

2020_11_21_201630_create_orders_table.php

Міграція створення таблиці замовлень.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateOrdersTable extends Migration
{
    public function up()
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->id();
            $table->integer('user_id');
            $table->integer('client_id');
            $table->text('comment')->nullable();
            $table->boolean('status')->default(0);
            $table->string('time')->nullable();
            $table->string('car');
            $table->text('services')->nullable();
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('orders');
    }
}
```

2020_11_21_201417_create_services_table.php

Міграція створення таблиці послуг.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateServicesTable extends Migration
{
    public function up()
    {
        Schema::create('services', function (Blueprint $table) {
            $table->id();
            $table->string('title');
```

```
            $table->boolean('selected')->default(0);
        });
    }
    public function down()
    {
        Schema::dropIfExists('services');
    }
}
```

2020_11_21_202449_create_user_has_services_table.php

Міграція створення таблиці послуг обраних користувачем.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUserHasServiceItemsTable extends Migration
{
    public function up()
    {
        Schema::create('user_has_service_items', function (Blueprint $table) {
            $table->id();
            $table->integer('user_id');
            $table->integer('service_item_id');
            $table->string('price')->nullable();
        });
    }
    public function down()
    {
        Schema::dropIfExists('user_has_service_items');
    }
}
```

2020_11_21_202216_create_reviews_table.php

Міграція створення таблиці відгуків.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateReviewsTable extends Migration
{
    public function up()
    {
        Schema::create('reviews', function (Blueprint $table) {
```

```
            $table->id();
            $table->integer('user_id');
            $table->integer('client_id');
            $table->text('comment');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('reviews');
    }
}
```

2020_11_21_202939_create_news_table.php

Міграція створення таблиці новин.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateNewsTable extends Migration
{
    public function up()
    {
        Schema::create('news', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->string('photo')->default('/img/no-image.png');
            $table->string('description');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('news');
    }}
```

User.php

Модель користувачів.

```
<?php

namespace App\Models;
```

```
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Passport\HasApiTokens;

class User extends Authenticatable
{
    use Notifiable, HasApiTokens;

    protected $fillable = [
        'name',
        'surname',
        'phone',
        'address',
        'photo',
        'service_name',
        'city_id',
        'user_role_id',
        'email',
        'password',
        'description'
    ];

    protected $hidden = [
        'password',
    ];
    function services() {
        return $this->HasMany('App\Models\UserHasServices', 'user_id');
    }
    function serviceItems() {
        return $this->HasMany('App\Models\UserHasServiceItems', 'user_id');
    }
    function photos() {
        return $this->HasMany('App\Models\UserPhoto', 'user_id');
    }
    function orders() {
        return $this->HasMany('App\Models\Orders', 'user_id');
    }
    function reviews() {
        return $this->HasMany('App\Models\Reviews', 'user_id');
    }
    function cars() {
        return $this->HasMany('App\Models\UserCars', 'user_id');
    }
}
```

Orders.php

Модель замовлень.

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Orders extends Model
{
    use HasFactory;

    protected $table = 'orders';

    protected $fillable = [
        'client_id',
        'comment',
        'status',
        'time',
        'car',
        'services',
        'user_id',
        'name',
        'phone',
        'email'
    ];

    function user() {
        return $this->belongsTo('App\Models\User', 'user_id');
    }
}
```

Services.php

Модель послуг.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Services extends Model
{
    use HasFactory;

    protected $table = 'services';
    public $timestamps = false;

    protected $fillable = [
        'title'
```

```
    ];

    function items() {
        return $this->hasMany('App\Models\ServiceItems', 'service_id');
    }
   function itemsHesUsers() {
        return $this->hasMany('App\Models\UserHasServiceItems', 'service_id');
    }
}
```

UserHasServiceItems.php

Модель послуг користувачів.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserHasServiceItems extends Model
{
    use HasFactory;

    protected $table = 'user_has_service_items';
    public $timestamps = false;

    protected $fillable = [
        'user_id',
        'service_item_id',
        'price'
    ];

    function item() {
        return $this->belongsTo('App\Models\ServiceItems', 'service_item_id');
    }
}
```

Reviews.php

Модель відгуків.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class Reviews extends Model
{
    use HasFactory;

    protected $table = 'reviews';
    protected $fillable = [
        'user_id',
        'client_id',
        'comment'
    ];
    function user() {
        return $this->belongsTo('App\Models\User', 'client_id');
    }
}
```

AuthController.php

Контроллер авторизації.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

use App\Models\User;

class AuthController extends Controller
{
    // register
    function register(Request $request) {
        $request->validate([
            'email' => 'required|string|email|unique:users',
            'password' => 'required|string'
        ]);
        $user = new User();
        $data = $request->all();
        $data["password"] = Hash::make($request->password);
        $user->create($data);
        $credentials = request(['email', 'password']);
        if(!Auth::attempt($credentials)) {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
        $authUser = Auth::user();
        $tokenResult = $authUser->createToken('Personal Access Token');
        $token = $tokenResult->token;
        $token->save();
        return response()->json([
```

```
            'access_token' => 'Bearer '.$tokenResult->accessToken,
            'user' => $authUser
        ]);
    }

    // login
    function login(Request $request) {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'user_role_id' => 1]) || Auth::attempt(['email' => $request->email, 'password' => $request->password, 'user_role_id' => 2]) || Auth::attempt(['email' => $request->email, 'password' => $request->password, 'user_role_id' => 3])) {
            $user = Auth::user();
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            if($request->remember_me) {
                $token->expires_at = Carbon::now()->addWeeks(1);
            }
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken,
                'user' => $user
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }
    // loginAdmin
    function loginAdmin(Request $request) {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id' => 3])) {
            $user = $request->user();
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }
}
```

UserController.php

Контроллер користувачів.

```
<?php

namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;

use App\Models\User;
use App\Models\UserHasServices;
use App\Models\UserHasServiceItems;
use App\Models\UserPhoto;
use App\Models\Reviews;
use App\Models\UserCars;

class UserController extends Controller
{
    protected $fileStorage = "userfiles/";

    //getPopularServices
    function getPopularServices() {
        $data = User::with(
            'services.service',
            'serviceItems.item',
            'photos'
        )->where("user_role_id", 2)->orWhere("user_role_id", 3)->limit(6)->get();
        return response()->json($data);
    }

    //postUserCar
    function postUserCar(Request $request) {
        $model = new UserCars();
        $data = $request->all();
        $data['user_id'] = Auth::id();
        $model->create($data);
        return response('ok', 200);
    }

    //delUserCar
    function delUserCar($id) {
        UserCars::find($id)->delete();
        return response('ok', 200);
    }

    //postReview
    function postReview(Request $request, $id) {
        $model = new Reviews();
        $model->create([
            "user_id" => $id,
            "client_id" => Auth::id(),
            "comment" => $request->comment
        ]);
        return response('ok', 200);
```

```
    }

    //delReview
    function delReview($id) {
        Reviews::find($id)->delete();
        return response('ok', 200);
    }

    //getServices
    function getServices(Request $request) {
        $data = User::with('services.service', 'serviceItems.item', 'photos')->where("user_role_id", 2)->orWhere("user_role_id", 3)->get();
        return response()->json($data);
    }
    //getServiceId
    function getServiceId($id) {
        $data = User::with('services.service', 'serviceItems.item', 'photos', 'reviews.user')->find($id);
        foreach ($data['reviews'] as $key => $value) {
            $value['date'] = Carbon::parse($value['created_at'])->format('d.m.Y');
        }
        return response()->json($data);
    }

    //profile
    function profile() {
        $data = User::with(
            'services.service',
            'serviceItems.item',
            'photos',
            'orders.user',
            'reviews.user',
            'cars'
        )->find(Auth::id());
        foreach ($data['orders'] as $key => $value) {
            $value['date'] = Carbon::parse($value['created_at'])->format('d.m.Y');
            $value['car'] = json_decode($value['car']);
            $value['services'] = json_decode($value['services']);
        }
        return response()->json($data);
    }

    // updateUser
    function updateProfile(Request $request) {
        $id = Auth::id();
        $data = $request->all();
        if(isset($data['newPassword'])) {
            $user = Auth::user();
            if (Hash::check($data['oldPassword'], $user->password)) {
                $data["password"] = Hash::make($request->newPassword);
            } else {
```

```
            return response('error', 401);
        }
    }
    if($request->services) {
        UserHasServices::where("user_id", $id)->delete();
        UserHasServiceItems::where("user_id", $id)->delete();
        foreach ($request->services as $key => $value) {
            $model = new UserHasServices();
            $model->create([
                "user_id" => $id,
                "service_id" => $value['id']
            ]);
            foreach ($value['items'] as $k => $v) {
                if($v['selected'] == 1) {
                    $modelItem = new UserHasServiceItems();
                    $modelItem->create([
                        "user_id" => $id,
                        "service_item_id" => $v['id'],
                        "price" => isset($v['price']) ? $v['price'] : null
                    ]);
                }
            }
        }
    }
    User::find($id)->update($data);
    return response()->json(User::with('services.service', 'serviceItems.item')->find($id));
}

// postProfilePhoto
function postProfilePhoto(Request $request) {
    $uploadedFiles = $request->pics;
    foreach ($uploadedFiles as $file) {
        $model = new UserPhoto();
        $puth = $this->fileStorage.Auth::id();
        $name = 'http://'.$_SERVER['HTTP_HOST'].'/'.$puth."/".uniqid().'.'.$file-
>getClientOriginalExtension();
        $file->move(public_path().'/'.$puth, $name);
        $model->create([
            "user_id" => Auth::id(),
            "src" => $name
        ]);
    }
    return response('ok', 200);
}
// delProfilePhoto
function delProfilePhoto($id) {
    UserPhoto::find($id)->delete();
    return response('ok', 200);
}
// img Account
```

```
    function img(Request $request) {
        if(isset($request['photo'])) {
            $arr = [];
            if($request['photo']) {
                $file = uniqid().'_photo_min.png';
                $uploadfile = $this->fileStorage . $request['id'] . '/' . $file;

                $img = str_replace('', '', $request['photo']);
                $img = str_replace(' ', '+', $img);
                $fileData = base64_decode($img);
                file_put_contents(public_path().'/'.$uploadfile, $fileData);

                $arr['status'] = 'success';
                $arr['path_mini'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
                $arr['file_mini'] = $file;
            }
        }
        else {
            if(!file_exists("userfiles/".$request['id'])) {
                mkdir($this->fileStorage.$request['id']);
            }
            $uploadfile = $this->fileStorage. $request['id'] . '/' . uniqid().'_photo_original.png';
            $arr = array();
         if (move_uploaded_file($_FILES['file']['tmp_name'], public_path().'/'.$uploadfile)) {
                $arr['status'] = 'success';
                $arr['path_max'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
                $arr['file_max'] = $_FILES['file']['name'];
            } else {
              $arr['status'] = 'fail';
            }
        }
        header('Content-type: application/json');
        return response()->json($arr);
    }
}
```

OrderController.php

Контроллер замовлень.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Orders;

class OrderController extends Controller
{
    // postOrder
```

```
    function postOrder(Request $request, $id) {
        $model = new Orders();
        $model->create([
            "comment" => $request->comment,
            "client_id" => Auth::id() ? Auth::id() : null,
            "time" => $request->time,
            "user_id" => $id,
            "name" => $request->name,
            "phone" => $request->phone,
            "email" => $request->email,
            "car" => json_encode($request->car),
            "services" => json_encode($request->services)
        ]);
        return response('ok', 200);
    }
    // updateOrder
    function updateOrder($id) {
        Orders::find($id)->update([
            "status" => 1
        ]);
        return response('ok', 200);
    }
    // delOrder
    function delOrder($id) {
        Orders::find($id)->delete();
        return response('ok', 200);
    }
}
```

api.php

Перелік запитів на сервер.

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

//auth
Route::post('register', 'App\Http\Controllers\AuthController@register');
Route::post('login', 'App\Http\Controllers\AuthController@login');
Route::post('login-admin', 'App\Http\Controllers\AuthController@loginAdmin');
Route::post('img', 'App\Http\Controllers\UserController@img');

//services
Route::get('services', 'App\Http\Controllers\ServiceController@get');
Route::get('services-items', 'App\Http\Controllers\ServiceController@getItems');

//regions
```

```
Route::get('regions', 'App\Http\Controllers\CityController@getRegions');
Route::get('city', 'App\Http\Controllers\CityController@getCity');

//cars
Route::get('cars', 'App\Http\Controllers\CarController@getCars');
Route::get('series', 'App\Http\Controllers\CarController@getSeries');

Route::group([
    'middleware' => 'auth:api'
], function() {
    // users
    Route::get('profile', 'App\Http\Controllers\UserController@profile');
    Route::post('profile', 'App\Http\Controllers\UserController@updateProfile');
    Route::post('profile/photo', 'App\Http\Controllers\UserController@postProfilePhoto');
    Route::post('profile/del-photo/{id}', 'App\Http\Controllers\UserController@delProfilePhoto');
    Route::get('service', 'App\Http\Controllers\UserController@getServices');
    Route::get('service/{id}', 'App\Http\Controllers\UserController@getServiceId');

    // orders
    Route::post('order/{id}', 'App\Http\Controllers\OrderController@postOrder');
    Route::post('del-order/{id}', 'App\Http\Controllers\OrderController@delOrder');
    Route::post('update-order/{id}', 'App\Http\Controllers\OrderController@updateOrder');

    // reviews
    Route::post('reviews/{id}', 'App\Http\Controllers\UserController@postReview');
    Route::post('del-reviews/{id}', 'App\Http\Controllers\UserController@delReview');

    // user car
    Route::post('user-car', 'App\Http\Controllers\UserController@postUserCar');
    Route::post('del-user-car/{id}', 'App\Http\Controllers\UserController@delUserCar');
    Route::get('popular-services', 'App\Http\Controllers\UserController@getPopularServices');
});
```

Index.vue

Компонент головної сторінки сайту.

```
<template>
    <div>
        <div class="header-bg"></div>
        <b-container>
            <div class="form-search">
                <h1 class="header">Почніть пошук</h1>
                <b-row class="inputs m-0">
                    <b-col>
                        <multiselect
                            :options="servicesItems"
                            v-model="search.services"
                            placeholder="Послуга"
```

```
            label="title"
            track-by="id"
            tagPlaceholder="Обрати"
            selectLabel="Обрати"
            noResult="Нічого не знайдено"
          ></multiselect>
        </b-col>
        <b-col>
          <multiselect
            :options="cars"
            v-model="search.cars"
            placeholder="Авто"
            label="title"
            track-by="id"
            selectLabel="Обрати"
            noResult="Нічого не знайдено"
          ></multiselect>
        </b-col>
        <b-col>
          <multiselect
            :options="city"
            v-model="search.city"
            placeholder="Місто"
            label="title"
            track-by="id"
            selectLabel="Обрати"
            noResult="Нічого не знайдено"
          ></multiselect>
        </b-col>
    </b-row>
    <button class="button">Пошук сервісу</button>
</div>
<div class="block-title">
    ЯК DREAMCAR МОЖЕ МЕНІ ДОПОМОГТИ?
</div>
<b-row>
    <b-col class="help-item">
        <img src="/img/item-1.png">
        <div class="title">
            Тільки у справі
        </div>
        <div class="description">
            Всі відгуки залишені клієнтами, які вже скористалися послугами.
        </div>
    </b-col>
    <b-col class="help-item">
        <img src="/img/item-2.png">
        <div class="title">
            Знаємо, що пропонуємо
        </div>
```

```
                <div class="description">
                    Ми стежимо за рівнем надання послуг і рекомендуємо перевірені нами компанії.
                </div>
            </b-col>
            <b-col class="help-item">
                <img src="/img/item-4.png">
                <div class="title">
                    Простий онлайн-запис
                </div>
                <div class="description">
                    Замовляєте послугу, отримуєте SMS з підтвердженням замовлення, а також SMS-нагадування про майбутній візит.
                </div>
            </b-col>
        </b-row>
        <div class="block-title">
            ПОПУЛЯРНІ СЕРВІСИ
        </div>
        <b-row>
            <b-col cols="4" v-for="(item, index) in popularServices" :key="index">
                <ServiceItem :user="item"></ServiceItem>
            </b-col>
        </b-row>
        <div class="block-title">
            ОСТАННІ НОВИНИ
        </div>
        <NewsItem></NewsItem>
        <div class="block-title">
            БІЛЬШЕ МОЖЛИВОСТЕЙ
        </div>
    </b-container>
    <div class="block-auth">
        <b-container>
            <div class="login">
                <div class="title">
                    Авторизація
                </div>
                <input v-model="credentials.email" type="text" placeholder="Email">
                <input v-model="credentials.password" type="password" placeholder="Пароль">
                <button @click="login">Вхід</button>
            </div>
        </b-container>
    </div>
  </div>
</template>
<script>
import { services } from "../../mixins/services";
import { cars } from "../../mixins/cars";
import { city } from "../../mixins/city";
import Multiselect from 'vue-multiselect';
```

```
import ServiceItem from "../../components/site/ServiceItem";
import NewsItem from "../../components/site/NewsItem";
export default {
    mixins: [services, cars, city],
    components: {
        ServiceItem,
        NewsItem,
        Multiselect
    },
    data() {
        return {
            loading: false,
            search: {
                services: "",
                cars: "",
                city: ""
            },
            credentials: {
                email: "",
                password: ""
            },
            popularServices: []
        }
    },
    created() {
        this.fetchServicesItems();
        this.fetchCars();
        this.fetchCity();
        this.getPopularServices();
    },
    methods: {
        getPopularServices() {
            axios.get('/api/popular-services')
            .then((response) => {
                this.popularServices = response.data;
            })
        },
        login() {
            this.loading = true;
            this.$store.dispatch('login', this.credentials)
            .then(() => {
                window.location.href = '/profile';
            })
            .catch(err => {
                this.loading = false;
            })
        },
    }
}
</script>
```

```
<style lang="css" scoped>
    .form-search {
        padding-top: 40px;
        min-height: 200px;
        width: 80%;
        background: #ffffff;
        margin: 0 auto;
        border-radius: 10px;
        margin-top: -100px;
    }
    .form-search .header {
        text-align: center;
        color: #555555;
        font-size: 24px;
        margin-bottom: 20px;
    }
    .form-search .button {
        width: 100%;
        color: #ffffff;
        font-weight: bold;
        background: #051F61;
        border: 0;
        padding: 20px 0;
        margin-top: 40px;
        border-bottom-right-radius: 10px;
        border-bottom-left-radius: 10px;
    }
    .form-search .inputs input {
        border-radius: 5px;
        background: #ffffff;
        width: 100%;
        padding: 10px;
        border: 0;
        box-shadow: 0px 2px 3px rgba(0, 0, 0, 0.2);
        outline: none;
    }
    .block-auth .login .title {
        text-align: center;
        font-size: 28px;
        color: #051F61;
        font-weight: normal;
        margin-bottom: 20px;
    }
    .block-auth .login input {
        background: #ffffff;
        border: 1px solid rgb(175, 175, 175);
        border-radius: 10px;
        display: block;
        outline: none;
        margin-bottom: 15px;
```

```
    padding: 10px;
    font-size: 16px;
    width: 100%;
}
.block-auth .login button {
    background: #051F61;
    color: #ffffff;
    width: 100%;
    padding: 10px;
    margin-top: 15px;
    border: 0;
    border-radius: 10px;
}
.block-auth .login {
    background: #ffffff;
    padding: 35px;
    float: right;
    box-shadow: 0px 3px 6px rgba(75, 81, 91, 0.15), 0px 1px 3px rgba(0, 0, 0, 0.15);
    border-radius: 10px;
    width: 350px;
}
.block-auth {
    height: 330px;
    background: url("/img/auth_bg.jpg") no-repeat;
    background-size: cover;
    display: flex;
    align-items: center;
}
.help-item {
    text-align: center;
    padding: 0 50px;
}
.help-item .title {
    color: #051F61;
    font-weight: normal;
    font-size: 20px;
    line-height: 23px;
    padding: 30px 0;
}
.help-item .description {
    color: #374754;
    font-weight: 300;
    font-size: 13px;
    line-height: 20px;
}
.block-title {
    text-align: center;
    font-size: 31px;
    line-height: 37px;
    color: #051F61;
```

```
    font-weight: normal;
    margin: 50px 0;
  }
  .header-bg {
    height: 673px;
    background: url('/img/home_bg.png') no-repeat;
    background-size: cover;
  }
</style>
```

Services.vue

Компонент налаштування послуг.

```
<template>
  <div>
    <b-row>
      <b-col cols="6" v-for="(item, index) in compareServices" :key="index">
       <div v-for="(i, index) in item" :key="index" class="checkbox-item">
          <b-form-checkbox
            v-model="i.selected"
            value="1"
            unchecked-value="0"
            @change="selectService(i)"
          >{{ i.title }}</b-form-checkbox>
        </div>
      </b-col>
    </b-row>
    <div v-for="(item, index) in selectServices" :key="index">
      <hr>
      <div class="title-item">{{ item.title }}</div>
      <b-row>
        <b-col cols="6" v-for="(item, index) in compareServiceItems(item.items)" :key="index">
          <div v-for="(i, index2) in item" :key="index2" class="checkbox-item">
            <b-form-checkbox
              v-model="i.selected"
              value="1"
              unchecked-value="0"
            >{{ i.title }} <input type="text" v-model="i.price" class="price"
placeholder="Ціна"></b-form-checkbox>
          </div>
        </b-col>
      </b-row>
    </div>
    <div class="buttons-profile">
      <b-button type="submit" variant="primary" @click="save">
        <span class="spinner-border spinner-border-sm"
          role="status"
          aria-hidden="true"
          v-if="loading"
```

```
            ></span>
            <span class="sr-only" v-if="loading">Loading...</span>
            Зберегти
          </b-button>
        </div>
      </div>
</template>
<script>
export default {
    data() {
        return {
            loading: false,
            services: [],
            user: {
                services: [],
                service_items: []
            }
        }
    },
    created() {
        this.fetchData();
        this.fetchServices();
    },
    computed: {
        compareServices() {
            var listServices = [];
            for (let i = 0; i < Math.ceil(this.services.length / (Math.round(this.services.length / 2))); i++){
                listServices[i] = this.services.slice((i * (Math.round(this.services.length / 2))), (i *
(Math.round(this.services.length / 2))) + (Math.round(this.services.length / 2)));
            }
            return listServices;
        },
        selectServices() {
            return this.services.filter(item => item.selected == 1);
        }
    },
    methods: {
        selectService(item) {
            if(item.selected == 0) {
                item.items.forEach(element => {
                    element.selected = 0;
                })
            }
        },
        selectAll(item) {
            if(item.selectAll == true) {
                item.items.forEach(element => {
                    element.selected = 0;
                })
                item.selectAll = false;
```

```
            } else {
                item.items.forEach(element => {
                    element.selected = 1;
                })
                item.selectAll = true;
            }
        },
        fetchServices() {
            axios.get('/api/services')
            .then((response) => {
                this.services = response.data;
                this.services.forEach(item => {
                    if(this.user.services.find(i => i.service_id == item.id)) {
                        item.selected = 1;
                    }
                });
            })
        },
        fetchData() {
            axios.get('/api/profile')
            .then((response) => {
                this.user = response.data;
            })
        },
        compareServiceItems(items) {
            items.forEach(element => {
                var test = this.user.service_items.find(i => i.service_item_id == element.id);
                if(test) {
                    element.selected = 1;
                    element.price = test.price;
                }
            });
            var listServices = [];
            for (let i = 0; i < Math.ceil(items.length / (Math.round(items.length / 2))); i++){
                listServices[i] = items.slice((i * (Math.round(items.length / 2))), (i *
(Math.round(items.length / 2))) + (Math.round(items.length / 2)));
            }
            return listServices;
        },
        save() {
            this.loading = true;
             axios.post('/api/profile', {
                services: this.services.filter(item => item.selected)
             }).then((response) => {
                this.user = response.data;
                this.loading = false;
            })
        }
    }
}
```

```
</script>
<style lang="css" scoped>
   .price {
      width: 50px;
      float: right;
      margin-left: 15px;
   }
   .checkbox-item {
      padding: 10px 0;
   }
   .title-item {
      color: #4B515B;
      font-size: 20px;
      line-height: 23px;
   }
   .selectAll {
      border: 0;
      outline: none;
      float: right;
      color: #196af7;
      background: #ffffff;
   }
</style>
```

app.js

Скрипт підключення головних модулів сайту.

```
require('./bootstrap');

import Vue from 'vue';
import router from "./routes";
import store from "./store";
import { BootstrapVue, IconsPlugin } from 'bootstrap-vue';
import VeeValidate, { Validator } from 'vee-validate';
import VueSilentbox from 'vue-silentbox'
import 'vue-multiselect/dist/vue-multiselect.min.css';
import HeaderComponent from './components/site/Header';
import FooterComponent from './components/site/Footer';

Vue.use(VueSilentbox);
Vue.use(BootstrapVue);
Vue.use(IconsPlugin);
Vue.use(VeeValidate);
Vue.prototype.$http = axios;

const token = localStorage.getItem('token');
if (token) {
   Vue.prototype.$http.defaults.headers.common['Authorization'] = token;
}
```

```
Vue.config.productionTip = false

const app = new Vue({
    el: '#app',
    components: {
        HeaderComponent,
        FooterComponent
    },
    store,
    router
});
```

store.js

Скрипт відправлення та отримання запитів від серверу на клієнтську частину.

```
import Vue from 'vue'
import Vuex from 'vuex'
Vue.use(Vuex)

export default new Vuex.Store({
    state: {
        status: '',
        token: localStorage.getItem('token') || '',
        tokenAdmin: localStorage.getItem('tokenAdmin') || '',
        user: JSON.parse(localStorage.getItem('user')) || null
    },
    mutations: {
        auth_request(state) {
            state.status = 'loading'
        },
        auth_user_success(state, token, user_data) {
            state.status = 'success'
            state.token = token
            state.user = user_data
        },
        auth_admin_success(state, token) {
            state.status = 'success'
            state.tokenAdmin = token
        },
        auth_error(state) {
            state.status = 'error'
        },
        logout(state) {
            state.status = ''
            state.token = ''
            state.user = ''
        },
```

```
    user_data(state, user){
      state.user = user
    },
  },
  actions: {
    login({commit}, user) {
      return new Promise((resolve, reject) => {
        commit('auth_request')
        axios({url: '/api/login', data: user, method: 'POST' })
        .then(resp => {
          const token = resp.data.access_token
          const user_data = resp.data.user
          localStorage.setItem('user', JSON.stringify(user_data))
          localStorage.setItem('token', token)
          axios.defaults.headers.common['Authorization'] = token
          commit('auth_user_success', token, user_data)
          resolve(resp)
        })
        .catch(err => {
          commit('auth_error')
          localStorage.removeItem('token')
          localStorage.removeItem('user')
          reject(err)
        })
      })
    },
    loginAdmin({commit}, user) {
      return new Promise((resolve, reject) => {
        axios({url: '/api/login-admin', data: user, method: 'POST' })
        .then(resp => {
          const token = resp.data.access_token
          localStorage.setItem('tokenAdmin', token)
          axios.defaults.headers.common['Authorization'] = token
          commit('auth_admin_success', token)
          resolve(resp)
        })
        .catch(err => {
          commit('auth_error')
          localStorage.removeItem('tokenAdmin')
          reject(err)
        })
      })
    },
    register({commit}, user) {
      return new Promise((resolve, reject) => {
        commit('auth_request')
        axios({url: '/api/register', data: user, method: 'POST' })
        .then(resp => {
          const token = resp.data.access_token
          const user_data = resp.data.user
```

```
                localStorage.setItem('user', JSON.stringify(user_data))
                localStorage.setItem('token', token)
                axios.defaults.headers.common['Authorization'] = token
                commit('auth_success', token, user)
                resolve(resp)
            })
            .catch(err => {
                commit('auth_error', err)
                localStorage.removeItem('token')
                localStorage.removeItem('user')
                reject(err)
            })
        })
    },
    logout({commit}) {
        return new Promise((resolve, reject) => {
            commit('logout')
            localStorage.removeItem('token')
            localStorage.removeItem('user')
            localStorage.removeItem('tokenAdmin')
            delete axios.defaults.headers.common['Authorization']
            resolve()
        })
    },
    user({commit}, user) {
        localStorage.setItem('user', JSON.stringify(user))
        commit('user_data', user)
    }
  },
  getters: {
    isLoggedIn: state => !!state.token,
    isLoggedInAdmin: state => !!state.tokenAdmin,
    authStatus: state => state.status,
    authUser: state => state.user,
  }
})
```