

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ**

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

на тему: **РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО
СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

Фідірко Тимур Русланович

Керівник:

Толбатов Андрій Володимирович

к.т.н., доцент кафедри кібернетики та інформатики

Рецензент:

Дегтярьова Неля Валентинівна

к.п.н., доцент, завідувачка кафедри інформатики Сумського ДПУ
ім.А.С.Макаренка

Суми – 2024

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	«Бакалавр»
Спеціальність	<u>126 «Інформаційні системи та технології»</u> <u>ОП «Інформаційні системи та технології»</u>

Затверджую:
Завідувач кафедри Світлана АГАДЖАНОВА

«_____» «_____» 202_ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Фідирко Тимуру Руслановичу

1. Тема роботи: Розробка вебзастосунку обліку технічного стану легкових автомобілів

2. База практичного дослідження: ФОП «Флягін Андрій Сергійович», м. Суми
керівник роботи Толбатов Андрій Володимирович к.т.н., доцент
затверджена наказом по університету від «_____» _____ №_____

3. Строк подання студентом закінченого проекту (роботи) «28» червня 2024 року

4. Вихідні дані до проекту (роботи):

Результати аналітичних досліджень зарубіжних та вітчизняних вчених, інтернет-джерела, нормативні документи за чинним законодавством.

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

- виконати постановку задачі;
- вибрати середовище для реалізації;
- розробити технічне завдання;
- розробити програмний продукт відповідно до поставлених задач;
- розробити інструкцію для користувача.

6. Дата видачі завдання: «_____» _____ 202_ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проекту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	січень 2024 р.	виконано
2	Підбір та вивчення літературних джерел, технічної документації	лютий 2024 р.	виконано
3	Розробка технічного завдання	березень 2024 р..	виконано
4	Розробка вебзастосунку	лютий-квітень 2024 р.	виконано
5	Створення інструкції користування вебзастосунком для користувачів	до 10 травня 2024 р.	виконано
6	Оформлення пояснювальної записки	до 25 травня 2024 р..	виконано
7	Оформлення презентації	до 1 червня 2024 р.	виконано
8	Опрацювання зауважень керівника, перевірка на автентичність	4 червня 2024 р.	виконано
9	Підготовка доповіді та її попередній захист	10 червня 2024 р.	виконано
10	Перевірка з академічної доброчесності кваліфікаційної роботи	10 червня 2024 р.	виконано
11	Представлення кваліфікаційної роботи на кафедрі, деканат	14 червня 2024 р.	виконано
12	Захист кваліфікаційної роботи	28 червня 2024р.	

Студент

(підпис)

Тимур ФІДІРКО

Керівник роботи

(підпис)

Андрій ТОЛБАТОВ

Перевірку на автентичність проведено

(підпис)

Надія БАРАННІК

Кваліфікаційна робота допущена до захисту

(підпис)

Маргарита ЛИШЕНКО

АНОТАЦІЯ

Фідірко Т.Р. Розробка вебзастосунку обліку технічного стану легкових автомобілів.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології Сумського національного аграрного університету. Суми, 2024 р. – Рукопис.

Кваліфікаційна робота присвячена розробці вебзастосунку для обліку технічного стану легкових автомобілів із аналітичною підсистемою підбору станції технічного обслуговування (СТО). У роботі розглядаються основні методи та засоби автоматизації процесу збору та обробки даних для комплексного обслуговування автомобілів, аналізуються сучасні інформаційні системи та технології, що застосовуються в цій сфері. Вебзастосунок призначений для забезпечення зручного та ефективного інтерфейсу користувача, а також для автоматизації збереження та обробки даних.

Об'єктом дослідження є процеси надання комплексних послуг із технічного обслуговування автомобілів та вибору СТО. Метою роботи є розробка ефективного вебзастосунку, який дозволяє зручно збирати, зберігати та аналізувати дані про технічний стан легкових автомобілів.

Проведено проектування інформаційної системи, продемонстровано хід реалізації та приклади програмного коду. Після реалізації проведено тестування та демонстрація роботи функцій системи.

Результатом роботи є створення веборієнтованої інформаційної системи комплексного обслуговування автомобілів.

Ключові слова: інформаційна система, СТО, авто, ремонт, вебзастосунок, інформаційні системи, бази даних.

SUMMARY

Fidirko T.R. Development of a web application for accounting for the technical condition of passenger cars.

Qualification work on specialty 126 «Information systems and technologies» OP Information systems and technologies of the Sumy National Agrarian University. Sumy, 2024 - Manuscript.

The qualification work is devoted to the development of a web application for accounting for the technical condition of passenger cars with an analytical subsystem for selecting a service station (STO). The work examines the main methods and means of automating the process of data collection and processing for comprehensive car maintenance, analyzes modern information systems and technologies used in this field. The web application is designed to provide a convenient and efficient user interface, as well as to automate data storage and processing.

The object of the study is the process of providing complex services for car maintenance and choosing service stations. The purpose of the work is to develop an effective web application that allows you to conveniently collect, store and analyze data on the technical condition of passenger cars.

The design of the information system was carried out, the implementation process and examples of the software code were demonstrated. After the implementation, testing and demonstration of the system functions was carried out.

The result of the work is the creation of a web-oriented information system of comprehensive car maintenance.

Keywords: information system, service station, auto, repair, web application, information systems, databases.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Аналіз процесу вибору станцій технічного обслуговування автомобілів	9
1.2 Аналіз наявних послуг на станціях технічного обслуговування	11
1.3 Перспективи використання вебтехнологій для розробки інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО	12
РОЗДІЛ 2 ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ	15
2.1 Визначення мети та постановка задачі	15
2.2 Вибір засобів реалізації	16
РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ	20
3.1 Створення та проектування бази даних	20
3.2 Моделювання Use Case діаграми	27
3.3 Програмна реалізація вебзастосунку	30
ВИСНОВКИ	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	52
ДОДАТКИ	55
Додаток А Ідентифікація мети проекту методом SMART	
Додаток Б Планування робіт	
Додаток В Копії публікацій	
Додаток Г Лістинги програм	

ВСТУП

Актуальність теми. Людині потрібно не тільки купити автомобіль, але й подумати про його своєчасне технічне обслуговування. Багато людей нехтують цим питанням і продовжують їздити на автомобілях, які потребують належного обслуговування та модернізації, що є досить небезпечним. Така ситуація зумовлена різними причинами, такими як безвідповідальність і недбалість, а також низький рівень доходів населення. Заміна мастила, охолоджуючої рідини, заміна шин з літніх на зимові тощо – це те, що автовласники змушені робити самостійно.

Станції технічного обслуговування стали прибутковим бізнесом і легкодоступні майже в будь-якій точці світу завдяки власній дистрибуції. З такою великою кількістю варіантів автовласникам іноді важко вибрати найкращий сервіс з точки зору ціни, якості та розташування. При цьому в Інтернеті є багато сайтів, на яких можна знайти інформацію про доступні станції технічного обслуговування. Багато з них рекламують свої переваги, а деякі навіть дозволяють шукати СТО за типом діяльності, але вони не мають функціоналу для пошуку найкращого варіанту на основі обраних користувачем критеріїв.

Мета і завдання дослідження. Метою роботи є розробка вебзастосунку для обліку технічного стану легкових автомобілів, а також комплексного обслуговування транспортних засобів з аналітичною підсистемою підбору СТО для пошуку найкращого рішення за заданими користувачем критеріями.

Об'єктом дослідження є процеси надання комплексних послуг із технічного обслуговування автомобілів та вибору СТО. Метою роботи є розробка ефективного вебзастосунку, який дозволяє зручно збирати, зберігати та аналізувати дані про технічний стан легкових автомобілів.

Предметом дослідження є автоматизація реєстрації даних обслуговування автомобілів та вибору СТО.

Апробація результатів кваліфікаційної роботи. Результати кваліфікаційної роботи апробовані на конференціях: 1) всеукраїнської наукової конференції студентів і аспірантів, присвяченої міжнародному дню студента (13-

17 листопада 2023 р.), Суми; 2) науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ (14-16 травня 2024 р.).

Публікації. За матеріалами кваліфікаційної роботи опубліковано 2 тези наукових конференцій.

Структура і обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 31 найменування, додатків. Основний текст викладений на 54 сторінках комп'ютерного тексту, робота містить 1 таблицю, 43 рисунків, 4 додатки.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз процесу вибору станцій технічного обслуговування автомобілів

Роль автомобіля в сучасному суспільстві неможливо недооцінити. Важко уявити, що людство досягло такого стрімкого прогресу за останні 100 років. Автомобілі – це дуже складні механізми, що містять безліч систем і пристроїв. З часом виробники автомобілів вдосконалюють свої моделі, але, незважаючи на їхню якість і довговічність, будь-який механізм з часом може вийти з ладу. Сучасні транспортні засоби є високо технологічними. Тому діагностика несправного механізму або системи вимагає спеціального обладнання, яке є тільки на станціях технічного обслуговування. Кожен автовласник хоче довірити свій транспортний засіб в руки кваліфікованого фахівця з технічного обслуговування. При виборі станції технічного обслуговування слід звернути увагу на наступні моменти:

- перелік пропонованих послуг;
- наявність складу або магазину запасних частин
- якість пропонованих послуг і дотримання термінів виконання робіт;
- наявність додаткових послуг.

Слід зазначити, що хороші автосервіси не приховують інформацію від своїх клієнтів і ведуть бізнес відкрито.

При виборі також слід враховувати наступні додаткові міркування [1]

- досвід(якщо компанія працює на ринку давно, то, ймовірно, в ній працюють досвідчені фахівці);
- хороші рекомендації (незалежно від того, чи це рекомендації від знайомих, чи відгуки з професійних джерел);
- прийнятна ціна (вартість робіт не повинна бути занадто дешевою або занадто дорогою).

Наразі існує кілька типів автосервісів, які відрізняються за розміром, типом та якістю послуг. До них відносяться наступні:

- авторизовані дилери - це великі технічні центри, які працюють безпосередньо з виробниками транспортних засобів. Автодилерів іноді також називають роздрібними дистриб'юторами. На відміну від інших типів франчайзі, включаючи деяких дистриб'юторів, дилери рідко пропонують єдину лінійку продуктів [2]. Такі сервіси використовують лише оригінальні запчастини, а їхні працівники мають досвід і навички, щоб впоратися з будь-яким видом ремонту. Однак недоліком таких станцій є висока вартість обслуговування;

- незалежні сервіси можуть бути спеціалізованими (спеціалісти працюють з невеликою кількістю марок автомобілів) або універсальними (працівники працюють з усіма марками автомобілів і мають великий досвід). Вартість обслуговування на таких станціях значно нижча, ніж на офіційних СТО;

- приватні майстерні часто розташовані в гаражах. Ремонт тут може бути небезпечним. Характерною особливістю таких майстерень є те, що вони значно дешевші. Однак і серед таких майстрів трапляються хороші фахівці з великим досвідом роботи [3].

Автовласникам також варто звернути увагу на різні фактори [4], які визначають сумнівну станцію технічного обслуговування.

- грубі працівники: Солідні компанії не беруть на роботу працівників, які не ввічливі з клієнтами;

- примус споживачів до купівлі запчастин: Змушувати споживачів купувати запчастини або купувати лише витратні оригінальні матеріали чи матеріали тієї самої марки - це неповага. Обмежуючи свободу вибору споживачів, СТО намагаються заробити більше грошей та уникнути зайвого клопоту. При цьому ігноруються побажання та можливості власників транспортних засобів;

- умови праці працівників СТО: необхідно проаналізувати умови праці та звернути увагу на мікроклімат у приміщеннях;

- обмеження: деякі майстерні не мають кімнат відпочинку, що виходять вікнами на майстерню, або клієнтам заборонено перебувати в ремонтній зоні за

будь-яких обставин. Не варто довіряти автопрофесіоналам, які приховують щось від автовласника;

- примус до послуг замість рекомендацій: при діагностиці несправності автомобіля не всі клієнти можуть відрізнити пораду експерта від бажання примусити придбати більше товарів та послуг.

Проведений аналіз дозволяє зробити висновок, що існує багато факторів, які впливають на вибір станції технічного обслуговування. Це особливо актуально, якщо доводиться вручну шукати інформацію про кожну станцію технічного обслуговування.

1.2 Аналіз наявних послуг на станціях технічного обслуговування

Сервісні та ремонтні майстерні можна розділити на дві основні категорії за типом роботи, яку вони виконують. Перша – це універсальні майстерні, які пропонують всі види послуг. Інша-спеціалізовані майстерні. Вони пропонують лише певні види послуг.

Основні види робіт, які виконують станції технічного обслуговування, такі:

- поточне технічне обслуговування;
- кузовний ремонт;
- обслуговування або ремонт двигуна;
- обслуговування або ремонт трансмісії;
- ремонт ходової частини;
- несправність розвалу-сходження;
- ремонт електрики;
- обслуговування газобалонного обладнання;
- діагностика автомобіля.

Також можуть бути доступні наступні додаткові послуги:

- обслуговування або ремонт кондиціонера;
- шиномонтаж;
- встановлення мультимедійних систем;

- тюнінг та стайлінг автомобіля.

Як видно з наведеного вище, спектр доступних послуг з технічного обслуговування транспортних засобів є широким. Тому необхідно розробити інформаційну систему, яка б здійснювала пошук відповідних станцій технічного обслуговування за критеріями користувача (головним чином, за типом робіт, які необхідно виконати) і пропонувала найкращий вибір серед наявних. Це дозволить скоротити час і зусилля, необхідні клієнтам для вибору станції технічного обслуговування, яка відповідає їхнім можливостям і уподобанням.

1.3 Перспективи використання вебтехнологій для розробки інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО

Сьогодні багато аспектів сучасного життя переміщуються в інформаційні мережі, прискорюючи суспільний розвиток і долаючи географічні бар'єри. Інтернет пропонує людству широкі комунікаційні можливості за допомогою веб-сторінок, які є логічно пов'язаним набором програмних, медійних та інформаційних інструментів [5, 6, 7].

У сучасному бізнес-середовищі майже кожна організація має власне представництво в Інтернеті. Це є відображенням її успішності та конкурентоспроможності. Переваги використання вебсайту полягають у наступному:

- вебсторінки коштують дешево і їх легко створити: За допомогою веб-сайтів компанії можуть скоротити свої витрати. Це особливо актуально для ІТ-відділів, відповідальних за встановлення програмного забезпечення та подальшу підтримку. У цьому випадку користувачам потрібен лише комп'ютер або смартфон з підключенням до Інтернету;
- оновлення дешевші і менш складні: вартість підтримки програмного забезпечення так само важлива, як і його розробка. Регулярні оновлення є запорукою довговічності програмних продуктів. З огляду на це, варто зазначити,

що переваги, про які йшлося раніше, можна застосувати і до цієї ситуації. Додатки оновлюються лише на сервері, що дозволяє всім клієнтам одночасно працювати з новою версією програмного продукту;

- вебдодаток є більш універсальним і практичним. Зацікавлені сторони просто встановлюють вебдодаток на сервер під управлінням найновішої операційної системи (ОС). Клієнти можуть отримати доступ до нього через Інтернет з різних пристроїв, що працюють під управлінням MacOS, Windows, Android або інших операційних систем. Якщо вебдодаток розроблений якісно з використанням новітніх технологій розробки, він буде однаково добре працювати в будь-якому браузері;

- вебдодаток полегшує процес зберігання даних. Якщо до одних і тих же даних потрібно отримати доступ з різних місць, простіше об'єднати їх зберігання в одному місці. В результаті відпадає потреба в синхронізації.

Крім того, інтерактивні веб-сайти сьогодні мають можливість підтримувати комунікацію між адміністрацією та відвідувачами в режимі реального часу без додаткових витрат. Для цього існують онлайн-консультації, онлайн-чати (системи, що дозволяють обмінюватися миттєвими повідомленнями на сайті) та інші подібні інструменти.

Інтерактивні сайти для бізнесу дозволяють їх власникам отримувати додатковий дохід за рахунок активного залучення клієнтів, а відвідувачам-можливість користуватися зручними, якісними сервісами, що однозначно вигідно вирізнятиме відповідну організацію серед конкурентів.

Тенденції у відображенні вебданих. На початку розвитку веб-геопросторового картографування дані та карти створювалися фахівцями (наприклад, геодезія, фотограмметрія, дистанційне зондування) і були невеликими за розміром.

Значну частину цих даних складають геопросторові зображення та об'єкти. Наприклад, обсяг зображень Google Earth збільшився з приблизно 150 ТБ у 2006 році до 3 петабайт у 2016 році, включаючи повітряні, супутникові, 2D, 3D та історичні зображення [8]. Стрімке збільшення обсягів даних втягнуло сучасне

суспільство в епоху великих даних і всіх викликів, які вона несе[9]. Європейське космічне агентство доповнює велику кількість знімків у вільному доступі власними відкритими даними Sentinel [10], тоді як деякі приватні компанії також докладають чимало зусиль для створення супутникових даних [11]. PlanetScope [12] наразі експлуатує 149 супутників на орбіті і здатна щодня знімати всю поверхню суші Землі. Враховуючи лише оптичні супутникові знімки, Sentinel виробляє 2 терабайти даних на день, тоді як PlanetScope - 60 терабайт даних на день [13].

Google Maps API-це потужний інструмент, який можна використовувати для створення власних карт, карт з можливістю пошуку, функцій реєстрації, відображення даних у реальному часі, синхронізованих з даними про місцезнаходження, планування та створення маршрутів тощо.

Хоча Google Maps API дуже потужний, поєднання декількох API може дати ще кращі результати [14]. Таким чином, веб-візуалізація даних є дуже перспективним напрямком для комплексної інформаційної системи технічного обслуговування транспортних засобів, що розробляється.

У світлі попередніх пунктів, однією з головних переваг вебсайту є те, що ним можна користуватися з будь-якого пристрою, який підтримує доступ до Інтернету, незалежно від операційної системи тощо. Варто також зазначити, що технічна підтримка таких інформаційних систем є набагато простішою та менш затратною. Тому було вирішено, що інформаційна система, яка розробляється, має бути веб-орієнтованою з аналітичною підсистемою для вибору СТО.

Вона може бути синхронізована з інформацією про місцезнаходження користувача через геоприв'язку за допомогою Google Maps API. В результаті, пошук найбільш підходящої станції технічного обслуговування може здійснюватися на основі місцезнаходження клієнта.

РОЗДІЛ 2

ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Визначення мети та постановка задачі

Наразі в Інтернеті є багато вебсайтів автозаправних станцій. Якщо ви подивитесь і вивчите кожен з них, ви знайдете ряд спільних рис, таких як інформація про їхні послуги, їхні сильні сторони і чому їх обрали, а не конкурентів. Деякі з них також включають відгуки клієнтів, щоб відобразити свою популярність серед автовласників. Переглянувши п'ять-десять таких сайтів, вкрай складно вибрати найбільш зручну та доступну станцію технічного обслуговування та отримати якісне обслуговування за короткий час. Звісно, існують також сервіси, де постачальник послуг обирається відповідно до типу робіт, які потрібно виконати. Однак для цього потрібно дослідити і вибрати СТО, зв'язатися з ними через сайт або по телефону, дізнатися їх графік роботи і тарифи.

Якщо отримана інформація не відповідає потребам клієнта, процес потрібно повторити. У результаті автовласники можуть замислитися, навіщо вони витрачають час на таку роботу. Станції технічного обслуговування повинні бути зацікавлені в тому, щоб спростити автовласникам процес пошуку та вибору відповідної майстерні, зробити себе більш конкурентоспроможними та залучити більше клієнтів. Для цього сервіс має бути зручним і зрозумілим для обох сторін.

Тому основною метою даного проекту є розробка комплексної системи для технічного обслуговування автомобілів з аналітичною підсистемою для підбору СТО, яка знаходить оптимальні рішення відповідно до заданих користувачем критеріїв.

Для успішного досягнення цієї мети необхідно виконати наступні завдання:

- аналіз цільових областей технічного обслуговування транспортних засобів та вибору станцій технічного обслуговування;
- визначення мети та завдань проекту;

- аналіз вимог до проекту;
- підготовка технічного завдання;
- робочий план;
- визначення технології, яка буде застосована;
- вибір та встановлення дизайну;
- проектування бази даних;
- розробка серверної частини;
- тестування системи;
- документація.

При створенні будь-якого програмного продукту враховуються певні функціональні вимоги:

- реєстраційний модуль (реєстрація в якості клієнта, тобто власника транспортного засобу, реєстрація на СТО, особистий кабінет);
- модуль адміністрування (управління користувачами, додавання, редагування та видалення даних);
- модуль СТО (пошук СТО, підготовка заявок на ремонт: всі СТО, що відповідають запиту користувача, оцінять роботу з точки зору часу та вартості і надішлють інформацію клієнту).

2.2 Вибір засобів реалізації

Локальний сервер Open Server: Open Server Panel - це серверна платформа, створена спеціально для веб-розробників. Цей програмний продукт повинен складатися з набору серверних програмних інструментів разом зі зручним, функціональним і добре продуманим інтерфейсом [15].

Мова програмування PHP. PHP є найбільшпоширеною мовою длявеброзробки. PHP маєдовгу історію та велику підтримку. Мова програмування також має відкритий вихіднийкод [16]. PHP буде знайома програмістам, які

працювали з такими мовами, як C++ або Pascal. Це пов'язано з синтаксисом, запозиченим з цих мов програмування [17].

Мова програмування JavaScript. Це дуже цікавий інструмент програмування. JavaScript використовується для наступних цілей:

- створення динамічних сторінок;
- JavaScript пов'язує всі блоки разом і створює так званий фундамент;
- JavaScript може динамічно перевіряти форми перед відправкою їх на сервер [18];

CSS – технологія визначення зовнішнього вигляду HTML-документів. CSS використовується розробниками для встановлення шрифтів, кольорів фону та розташування елементів на сторінці. Мета створення каскадної таблиці стилів – відокремити зовнішній вигляд документа від його змісту [19].

HTML – це мова розмітки, а не мова програмування. Як формат зберігання даних, HTML готує представлення даних документа для відображення в браузері; HTML розвивається, і його інструменти, ймовірно, будуть використовуватися в майбутньому [20].

PHP-фреймворк Laravel – це фреймворк для створення веб-додатків з виразним та елегантним синтаксисом. Завдяки тому, що розробники Symfony зробили компоненти фреймворку незалежними один від одного у другій версії, його можна легко дизасемблювати і на основі цих деталей створити новий Laravel [21]. Потужний інструмент для швидкої розробки з усіма необхідними компонентами, Laravel є простим у використанні та інноваційним фреймворком, що підходить для широкого кола завдань [22].

Vue.js, створений Еваном Ю., є прогресивним JavaScript-фреймворком з відкритим вихідним кодом для побудови користувацького інтерфейсу (UI) та односторінкових додатків. Фреймворк дозволяє розробникам інкрементально створювати користувацькі інтерфейси [23].

Крім того, Vue надає набір модулів, які реалізують сучасний підхід до розробки веб-додатків [24]. Сильною стороною Vue.js є те, що він чітко задокументований [25].

Програмне забезпечення PhpStorm компанії JetBrains включає в себе інструменти для розробки веб-додатків[26]. У процесі використання IDE можна знайти корисні комбінації клавіш та інші можливості, висунуті розробником[27].

До складу PhpStorm також входить наступне:

- автоматична генерація коду;
- генерація PHPDoc;
- автоматичне форматування коду;
- підказки (завершення коду);
- консоль;
- тип пошуку.

Git. система контролю версій коду. Створена для полегшення процесу розробки програмного забезпечення в командах. Наразі є найпопулярнішою у світі, завдяки Git у програмісти можуть обмінюватися проектами і таким чином допомагати один одному [28].

Переваги:

- Git має всі переваги використання VCS, які має Subversion;
- має простий процес шифрування;
- якщо сервер з основним репозиторієм «падає», ви можете зробити коміти на локальному сервері і чекати, поки сервер відновить роботу [29].

Недоліки:

- зосередженість команди Git на наборах змін, а не на файлах, може бути несподіванкою для користувачів, які звикли до файлових VCS, таких як SVN;
- оскільки для ідентифікації ревізій використовується хеш SHA1, команді доводиться мати справу з довгими рядками, а не короткими номерами версій, як у багатьох інших системах;
- час, необхідний для створення історії певного файлу, є значним порівняно з файлово-орієнтованими системами;
- час, необхідний для створення історії певного файлу, є значним порівняно з файлово-орієнтованими системами;

- система не може відстежувати порожні каталоги;
- деякі команди можуть поводитися неочікувано, що може призвести, зокрема, до неочевидних помилок [30].

Тому вирішено реалізувати комплексну інформаційну систему для технічного обслуговування автомобілів з аналітичною підсистемою для вибору СТО з використанням сучасних вебтехнологій. Мова програмування PHP та фреймворк Laravel відповідають за серверну частину, а JavaScript та фреймворк Vue.js - за клієнтську. За базу даних відповідає MySQL. Сучасна веброзробка неможлива без використання систем контролю версій. У цьому дослідженні для написання коду використовуються Git та PHPStorm IDE. Також було обрано розробку на локальному сервері OpenServer.

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ

3.1 Створення та проектування бази даних

Структурно-функціональне моделювання - це процес створення окремих процесів об'єкта автоматизації. Реалізується у вигляді так званих ієрархічних діаграм взаємозв'язків. Компоненти системи визначаються у фіксованій декомпозиції.

Першим кроком є створення загальної діаграми. Ця діаграма містить загальний опис системи.

Наступним кроком є конкретизація загальної діаграми.

IDEF0 – це графічна нотація, яка використовується для створення функціональної моделі та відображення структури і функціональності системи.

Функціональні моделі реалізуються в обраному програмному продукті Erwin Process Modeler.

Контекстні діаграми – це діаграми, на яких модельовані об'єкти представлені однорідними блоками з граничними стрілками. Вони відображають певні взаємозв'язки між об'єктами та їх оточенням.

Контекстна діаграма процесу даної інформаційної системи зображена на рисунку 3.1.

Ця нотація також має можливість поступово вимальовувати основний процес. Для цього створюється піддіаграма, яка охоплює туж область, що і основний процес. На піддіаграмі процес описується більш детально. На основі цієї нотації стрілки основного процесу переносяться на піддіаграми за допомогою граничних стрілок. Існують різні типи стрілок: "Вхід", "Вихід", "Механізм", "Управління".

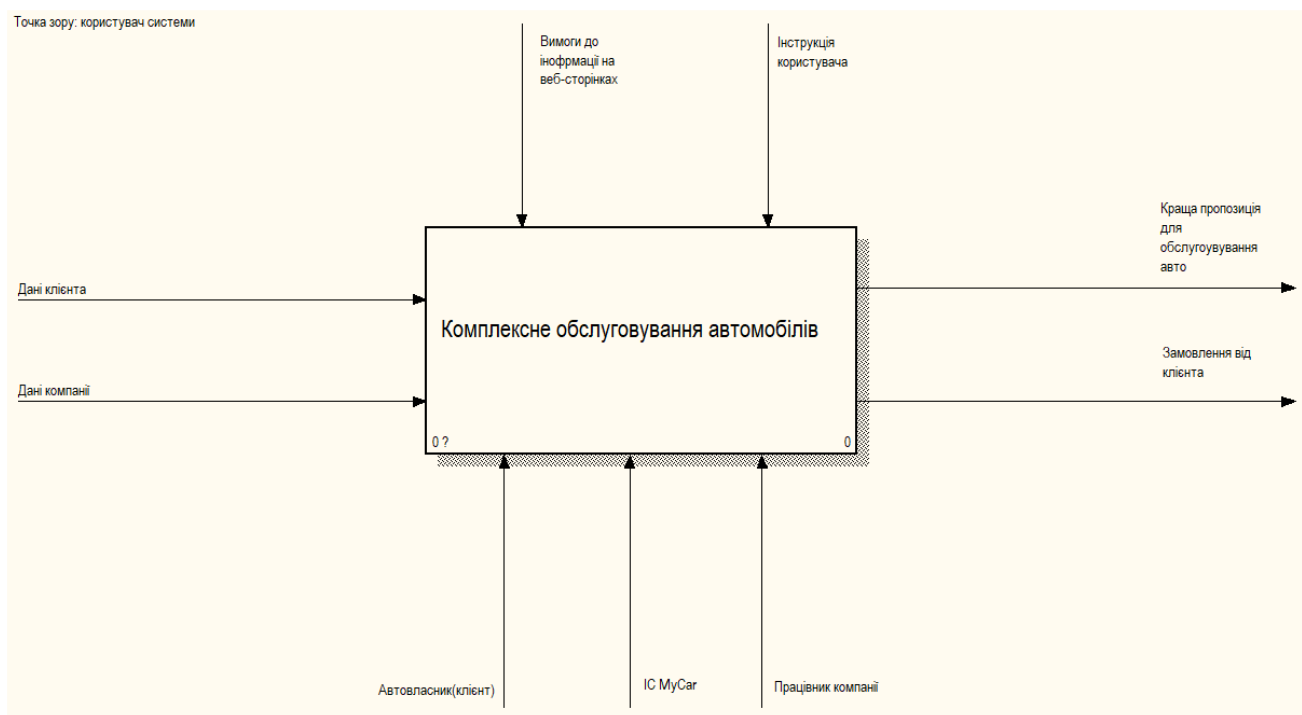


Рисунок 3.1 - Контекстна діаграма процесу

Джерело: запропоноване автором

Декомпозиція основної діаграми показана на рисунку 3.2.

Наступним кроком є більш детальне визначення процесів, які потребують конкретизації та детальної оцінки.

Виходи – це дані або матеріальні об'єкти, вироблені процесом. Входи перетворюються або споживаються процесом для створення того, що виробляється на виході.

Декомпозиція процесу авторизації та реєстрації показана на рисунку 3.3.

Розбивка процесу обробки заявок та вибору найкращої тендерної пропозиції (BOT) показана на рисунках 3.4-3.5.

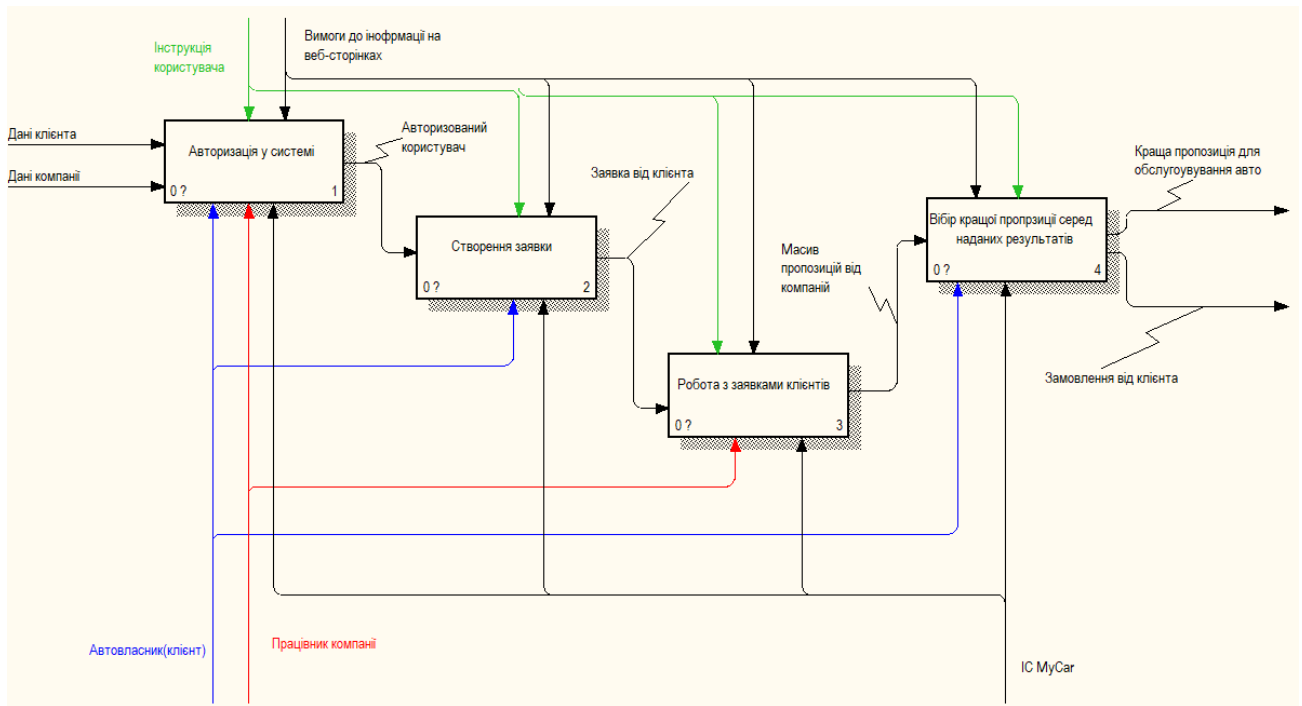


Рисунок 3.2 - Декомпозиція контекстної діаграми

Джерело: запропоновано автором

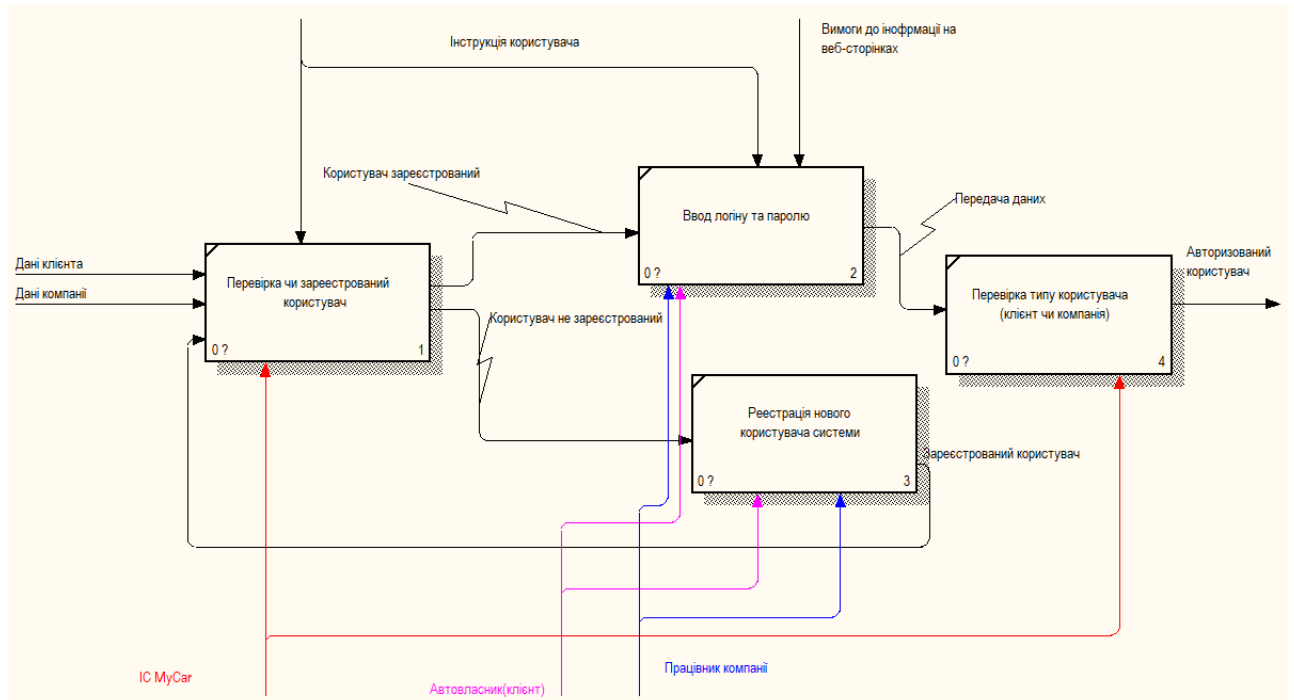


Рисунок 3.3 – Декомпозиція процесу авторизації

Джерело: запропоновано автором

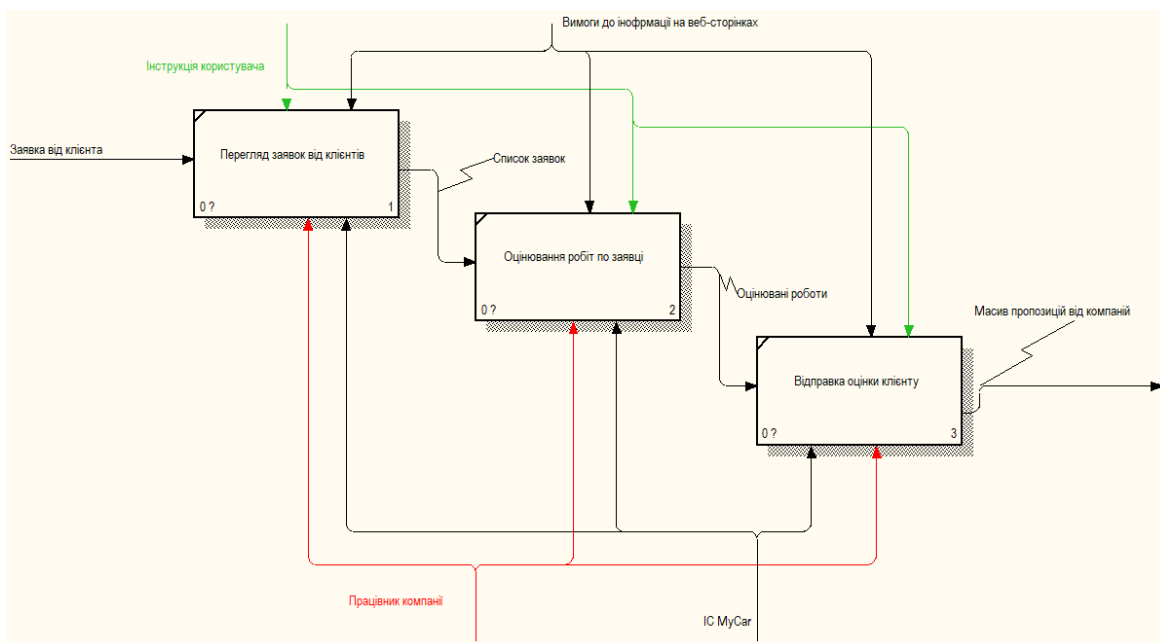


Рисунок 3.4 – Декомпозиція процесу роботи з заявками

Джерело: запропоновано автором

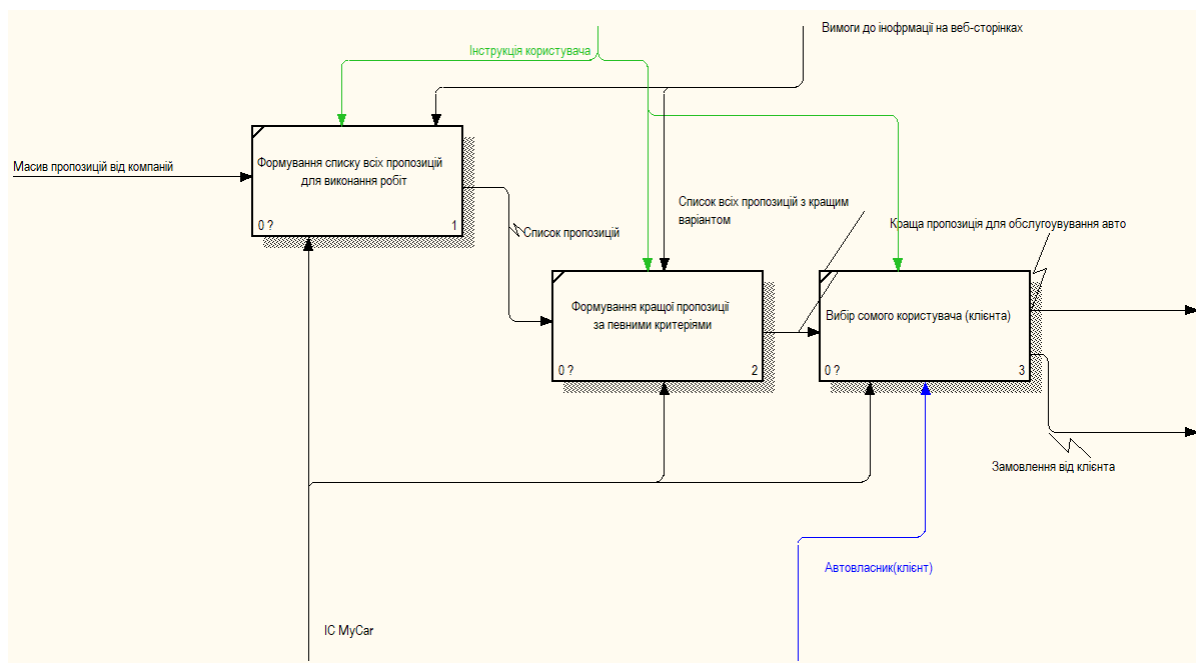


Рисунок 3.5 – Декомпозиція процесу вибору кращої пропозиції

Джерело: запропоновано автором

Механізми визначають засоби, які підтримують виконання процесу. Таким чином, блок IDEF0 показує перетворення від входу до виходу за допомогою механізму з урахуванням впливу контролю.

База даних складається з певної кількості даних з певної цільової області, що зберігаються в пам'яті.

Система управління базами даних (СКБД) повинна забезпечувати спеціальні механізми для пошуку даних за допомогою спеціальних, стандартизованих запитів і захищати дані від несанкціонованого доступу.

У реляційній моделі бази даних інформація зберігається у вигляді таблиць.

У базі даних можна створити велику кількість таблиць.

Це обмежується лише доступним обсягом пам'яті.

При роботі з декількома таблицями необхідно встановити певні зв'язки між ними.

При створенні бази даних необхідно детально розробити її структуру.

Цей процес включає наступні кроки:

- визначення теми, для якої створюється база даних;
- визначення конкретні таблиці, які повинна містити база даних;
- визначення списку полів для кожної таблиці;
- визначення ключових полів з унікальними значеннями;
- визначення зв'язків між таблицями;
- введення даних.

База даних повинна відповідати вимогам користувачів, які з нею працюють. Перш за все, слід визначити, які функції буде виконувати база даних і яку інформацію вона буде містити.

Схема даних розробленої системи зображена на рисунку 3.6.

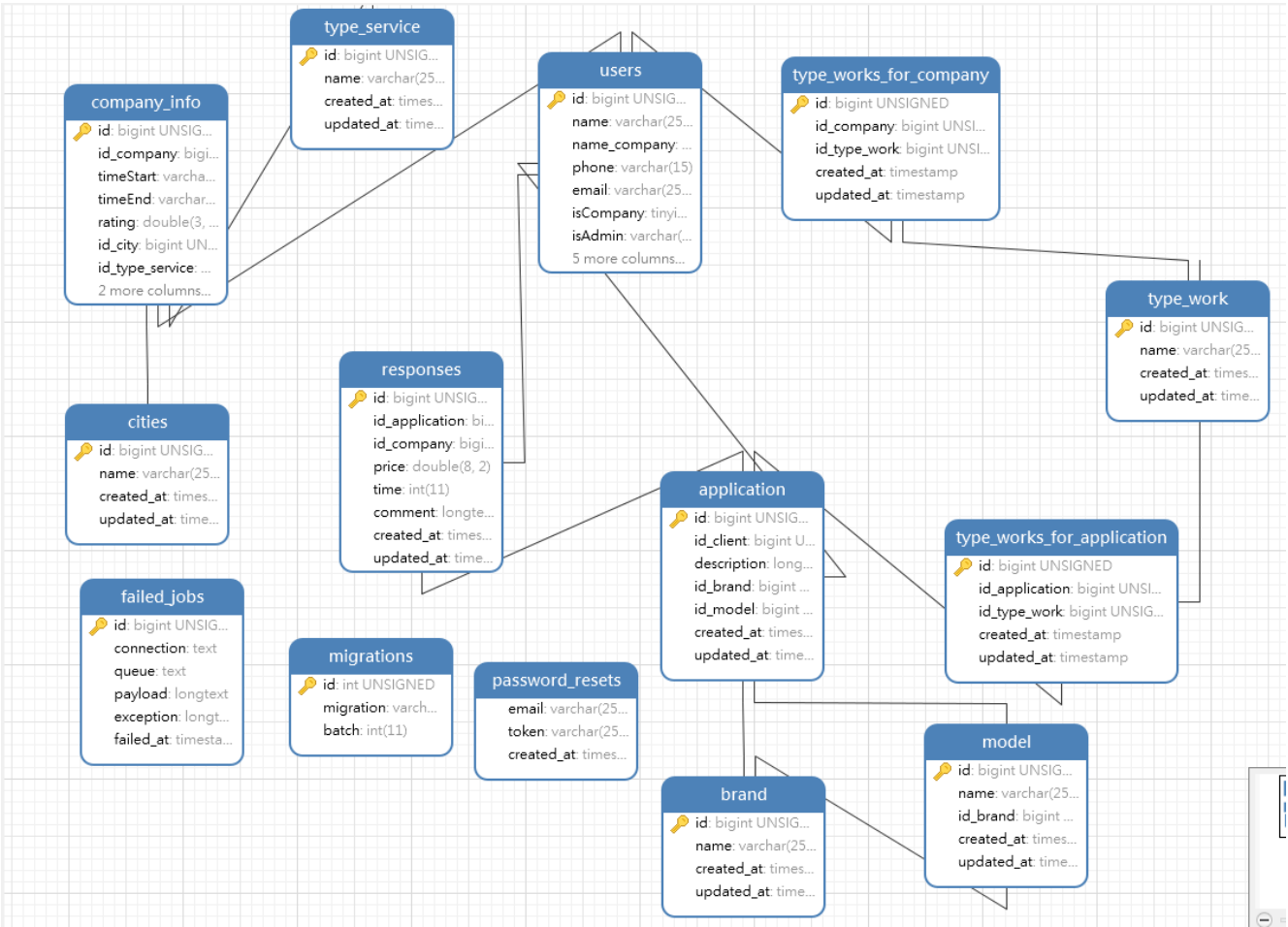


Рисунок 3.6 - Структура бази даних.
Джерело: запропоновано автором

Детальну інформацію про сутності та значення таблиць бази даних наведено в таблиці 3.1.

Таблиця 3.1 – Опис сутностей

Сутність	Тип даних	Значення
Таблиця Users		
id	bigint	Ідентифікаційний номер користувача
name	varchar	Ім'я клієнта
name_company	varchar	Назва компанії
phone	varchar	Телефон
email	varchar	Електронна пошта
isCompany	bigint	Ідентифікаційний номер користувача
isAdmin	varchar	Ім'я клієнта
password	varchar	Назва компанії

Продовження таблиці 3.1 – Опис сутностей

Сутність	Тип даних	Значення
Таблиця Users		
isCompany	bigint	Ідентифікаційний номер користувача
isAdmin	varchar	Ім'я клієнта
password	varchar	Назва компанії
Таблиця Cities		
id	bigint	Ідентифікаційний номер міста
name	varchar	Назва міста
Таблиця Company_info		
id	bigint	Ідентифікаційний номер запису
id_company	bigint	Ідентифікаційний номер компанії
id_city	bigint	Ідентифікаційний номер міста
id_type_service	bigint	Ідентифікаційний номер типу сервісу
Таблиця Type_work		
id	bigint	Ідентифікаційний номер запису
name	varchar	Назва типу робіт
Таблиця Type_works_for_company		
id	bigint	Ідентифікаційний номер запису
id_company	bigint	Ідентифікаційний номер компанії
id_type_work	bigint	Ідентифікаційний номер типу робіт
Таблиця brand		
id	bigint	Ідентифікаційний номер марки
name	varchar	Назва марки
Таблиця model		
id	bigint	Ідентифікаційний номер моделі
name	varchar	Назва моделі
Id_brand	bigint	Ідентифікаційний номер марки
Таблиця application		
id	bigint	Ідентифікаційний номер заявки
Id_client	bigint	Ідентифікаційний номер клієнта
description	varchar	Опис
Id_brand	bigint	Ідентифікаційний номер марки
Id_model	bigint	Ідентифікаційний номер моделі
Таблиця responses		
id	bigint	Ідентифікаційний номер пропозиції

Продовження таблиці 3.1 – Опис сутностей

Сутність	Тип даних	Значення
Таблиця responses		
Id application	bigint	Ідентифікаційний номер заявки
Id company	bigint	Ідентифікаційний номер компанії
price	double	ціна
time	int	час
comment	longtext	Коментар
Таблиця Type_works_for_application		
id	bigint	Ідентифікаційний номер запису
Id application	bigint	Ідентифікаційний номер заявки
id_type_work	Bigint	Ідентифікаційний номер типу робіт

Джерело: запропоновано автором

3.2 Моделювання Use Case діаграми

Діаграма варіантів використання або модель варіантів використання більш точно показує, як працює система. Діаграма опису є варіантом взаємодії між учасниками. Учасників може бути один або декілька. Учасником може бути людина або інша інформаційна система. Модель варіантів використання складається з опису учасників, варіанту використання (UC) для програмного продукту (SP) і діаграми варіантів використання. Діаграма варіантів використання для інформаційної системи (IC), що розробляється, показана на рисунку 3.7.

Дійові особи варіанту використання:

Адміністратор – може переглядати користувачів, редагувати їхні дані та видаляти їх за необхідності.

Власник автомобіля - клієнт, який створює запит на автосервіс, отримує всі можливі варіанти СТО (включаючи найкращий) відповідно до заданих критеріїв та обирає найкращий.

Співробітник СТО - співробітник компанії, який оцінює заявку на ремонт і відправляє її клієнту.

Варіанти використання:

ВВ Авторизація – дозволяє користувачам працювати з певним рівнем доступу та інтерфейсом.

ВВ Реєстрація – дозволяє користувачам реєструватися.

ВВ Редагування/видалення користувача – дозволяє системному адміністратору редагувати або видаляти дані користувача.

ВВ Створити заявку - дозволяє клієнту створити заявку на ремонт.

ВВ Оцінити заявку – дозволяє співробітнику отримати заявку, яка відповідає специфікаціям організації, оцінити роботу і відправити її заявнику.

ВВ Вибрати найкращий варіант – алгоритм системи для вибору найкращого варіанту на основі певних критеріїв.

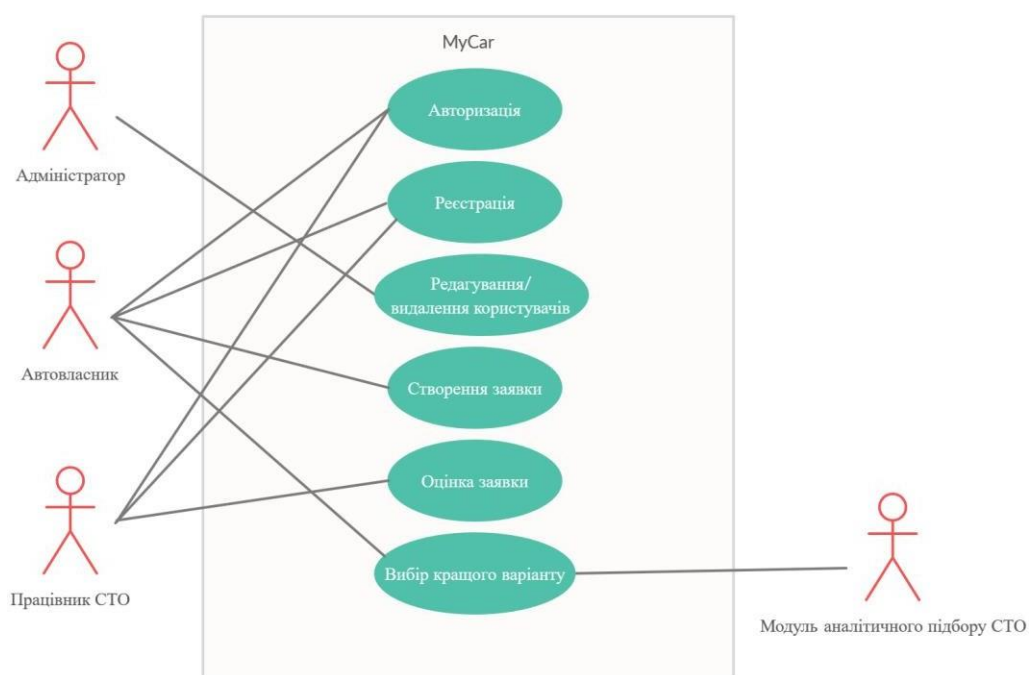


Рисунок 3.7 – Діаграма варіантів використання

Джерело: запропоновано автором

Враховуючи, що поставлено завдання вибрати найкращу пропозицію від СТО, можна зробити висновок, що на вході є масив, який містить пропозиції від СТО. Задача полягає в тому, щоб вибрати з цього масиву варіант, який найкраще відповідає наступним критеріям:

- вартість послуги (кожен хоче отримати найкращий результат за найнижчу ціну);

- час виконання робіт;

- рейтинг СТО.

Проаналізувавши наявні алгоритми, за основу було обрано Жадібний алгоритм, тому що цей алгоритм, який робить найкращий локальний вибір на кожному кроці з очікуванням, в очікуванні, що кінцеве рішення буде оптимальним.

На вхід подається послідовність усіх пропозицій для заданого попиту. Алгоритм пошуку найкращої пропозиції працює наступним чином:

- визначаються ваги критеріїв відповідно до проаналізованих даних;
- множина пропозицій ранжується за кожним критерієм та обирається найкраща;
- порівнюються результати.

Схема пошуку найкращої пропозиції показана на рисунку 3.8.



Рисунок 3.8 – Схема пошуку оптимальної пропозиції

Джерело: запропоновано автором

3.3 Програмна реалізація вебзастосунку

Згідно з завданням, система обліку технічного стану легкових автомобілів повинна бути розроблена у вигляді вебдодатку. Тому для початку реалізації необхідно налаштувати локальний вебсервер та базу даних. На платформах Windows найкращим вибором для розробки є OpenServer. Завдяки всім необхідним модулям та широкому спектру підтримуваних версій мови PHP, розробники мають комфортні умови для розробки програмних продуктів локально.

Дана система побудована на базі Laravel, яка є відмінним рішенням для швидкого і кваліфікованого створення безпечних і надійних вебдодатків, вона працює з використанням MVC, що дозволяє розробникам створювати вебдодаток за схемою сервера OpenServer. Ця схема показана на рисунку 3.9. Видно, що вона складається з декількох основних елементів:

- модель – взаємодіє з базою даних і надає необхідні дані;
- представлення – відображає дані, отримані від моделі та оброблені контролером;
- контролер – обробляє дані, отримані від моделі.

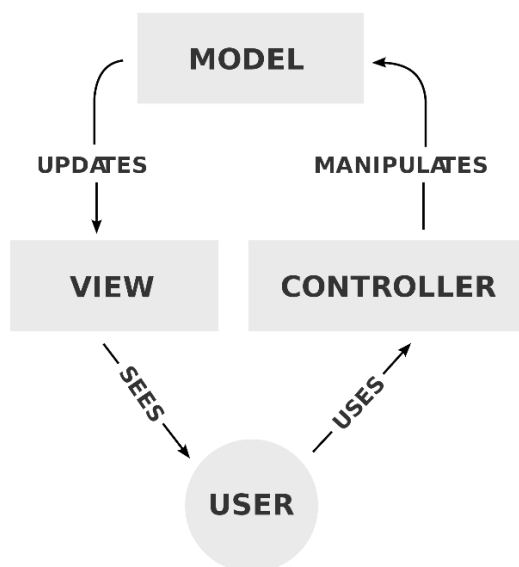


Рисунок 3.9 – Model – view – controller (MVC)

Джерело: [31]

Перед початком роботи з Laravel необхідно встановити Composer, який потрібен для встановлення необхідних моделей, в тому числі і самого Laravel. Щоб побачити Composer в дії, потрібно в командному рядку ввести команду `composer`, результат наведено на рисунку 3.10.

```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\wamp\www\laravel>php artisan --version
Laravel Framework version 5.1.23 (LTS)

C:\wamp\www\laravel>cd\

C:\>composer

Composer version 1.0-dev (c7ed232ef42c2bd63cdba057b6c7c8043b37cd5a) 2015-10-29 09:52:59

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                   Force ANSI output
  --no-ansi                Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                Display timing and memory usage information
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --verbose                Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
```

Рисунок 3.10 - Перевірка запуску Composer

Джерело: запропоновано автором

Якщо перевірка спрацювала, завантажується сам фреймворк:

`composer global require "laravel/installer"`

Composer завантажить необхідні файли Laravel до вибраного каталогу і після розархівування файлів необхідні каталоги стають доступними у вибраному каталозі, рисунок 3.11. Laravel має наступну структуру:

- `app` – каталог містить логічну структуру програми;
- `bootstrap` – каталог необхідний для запуску програми;
- `public` – каталог містить загальнодоступні файли, такі як точки входу та зображення;
- `vendor` – каталог є одним з найважливіших;
- містить власні файли Framework та додаткові бібліотеки, що використовуються.

.git	26.11.2020 22:22	Папка с файлами
.idea	27.11.2020 12:47	Папка с файлами
app	25.11.2020 22:29	Папка с файлами
bootstrap	22.09.2020 21:22	Папка с файлами
config	22.09.2020 21:22	Папка с файлами
database	22.09.2020 21:22	Папка с файлами
node_modules	14.11.2020 20:54	Папка с файлами
public	02.11.2020 23:34	Папка с файлами
resources	03.11.2020 22:09	Папка с файлами
routes	25.11.2020 23:44	Папка с файлами
storage	22.09.2020 21:22	Папка с файлами
tests	22.09.2020 21:22	Папка с файлами
vendor	01.11.2020 11:14	Папка с файлами
.editorconfig	22.09.2020 21:22	Файл ".EDITORCO...
.env	09.11.2020 23:57	Файл ".ENV"
.env.example	22.09.2020 21:22	Файл ".EXAMPLE"
.gitattributes	22.09.2020 21:22	Текстовый докум...
.gitignore	22.09.2020 21:22	Текстовый докум...
.styleci.yml	22.09.2020 21:22	Файл ".YML"
artisan	22.09.2020 21:22	Файл
composer.json	01.11.2020 11:08	Файл ".JSON"
composer.lock	01.11.2020 11:14	Файл ".LOCK"
package.json	14.11.2020 20:54	Файл ".JSON"
package-lock.json	14.11.2020 20:54	Файл ".JSON"
phpunit.xml	22.09.2020 21:22	Документ XML
README.md	22.09.2020 21:22	Файл ".MD"
server.php	22.09.2020 21:22	JetBrains PhpStorm
webpack.mix.js	01.11.2020 11:57	JetBrains PhpStorm

Рисунок 3.11 – Корневий каталог Laravel

Джерело: запропоновано автором

Для комфортної розробки вебдодатків потрібен зручний і функціональний текстовий редактор, наприклад PhpStorm (рисунок 3.12), 30-денна безкоштовна пробна версію знаходиться на офіційному сайті <https://www.jetbrains.com>.

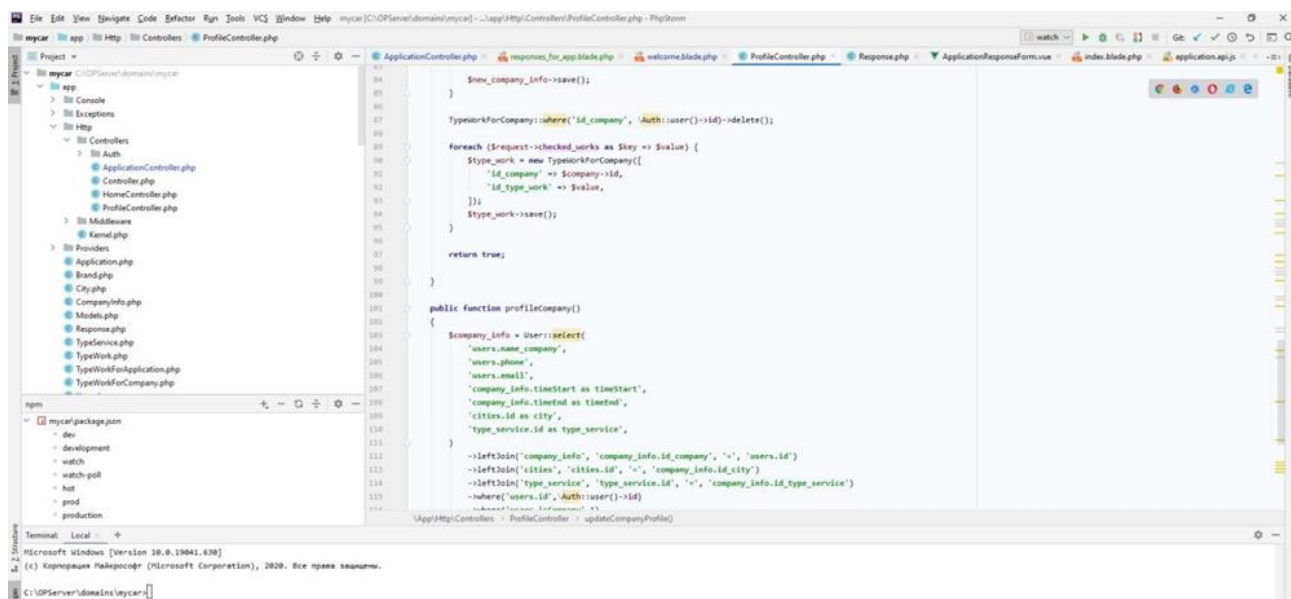


Рисунок 3.12 – Головне вікно PhpStorm

Джерело: запропоновано автором

Ситуації, коли вам потрібно скасувати зміни в коді або встановити попередню версію програми, цілком можливі, і Git був створений, щоб уникнути необхідності зберігати весь проект окремо кожного разу, коли ви вносите зміни. Це система для управління версіями коду.

Git встановлюється як звичайний додаток Windows. Інструкція по встановленню показана на рисунку 3.13 Git є консольним додатком і його незручно використовувати в консольному режимі. Тому для полегшення роботи з Git потрібно додатково встановити графічний додаток SmartGit (Рисунок 3.14).



Рисунок 3.13 - Git

Джерело: запропоновано автором

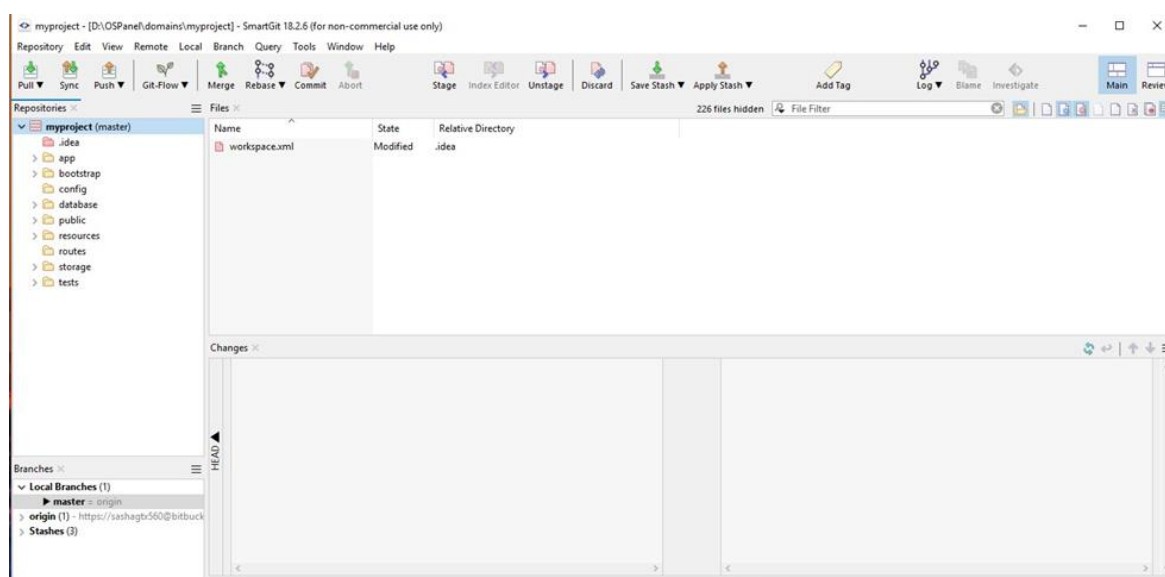


Рисунок 3.14 - SmartGit.

Джерело: запропоновано автором

Git зберігає дані у віддаленому сховищі. Найбільш поширеними онлайн-репозиторіями є GitHub та Bitbucket. Для розробки системи обрано Bitbucket, тому що їх функціональність майже ідентична (Рисунок 3.15).

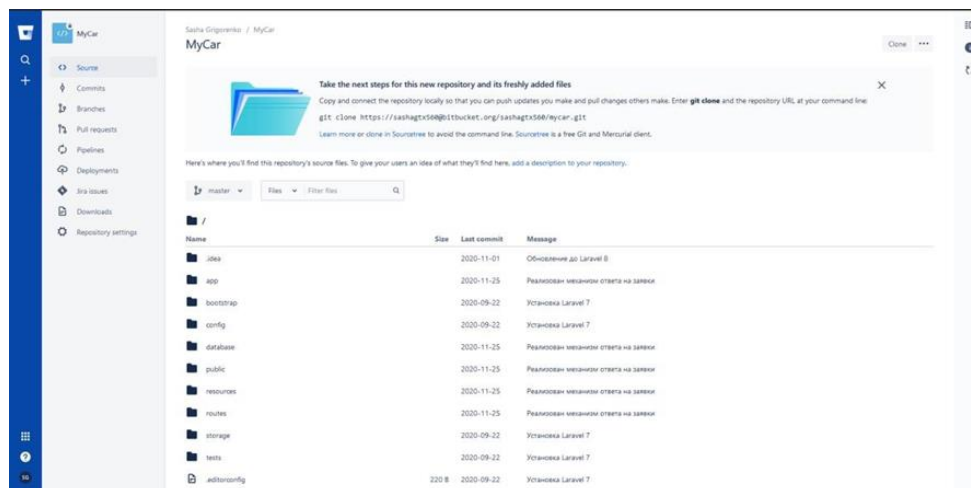


Рисунок 3.15 - Bitbucket

Джерело: запропоновано автором

Для коректної роботи Bitbucket та SmartGit потрібно підключити один до одного. Для цього спочатку створюють новий проект в Bitbucketi, а потім створюють копскапію SSH-ключа, який потрібно ввести у конфігурацію SmartGit, як показано на рисунку 3.16.

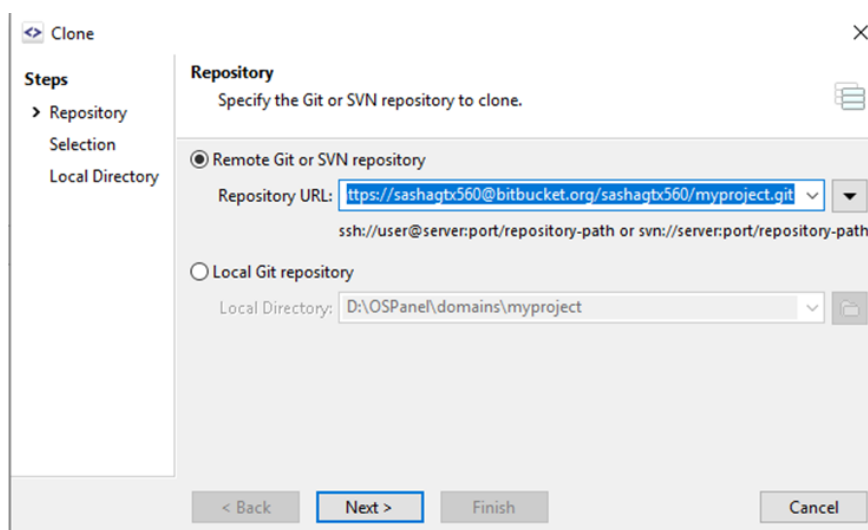


Рисунок 3.16 – Налаштування доступу SmartGit

Джерело: запропоновано автором

На кожному критичному етапі розробки слід додавати коментарі та відправляти їх у віддалений репозиторій Bitbucket за допомогою програми SmartGit. Це необхідно для того, щоб зміни в коді можна було в будь-який момент перевірити і, за необхідності, відкотитися до попередньої версії. Вікно коментарів, створених в SmartGit, показано на рисунку 3.17.

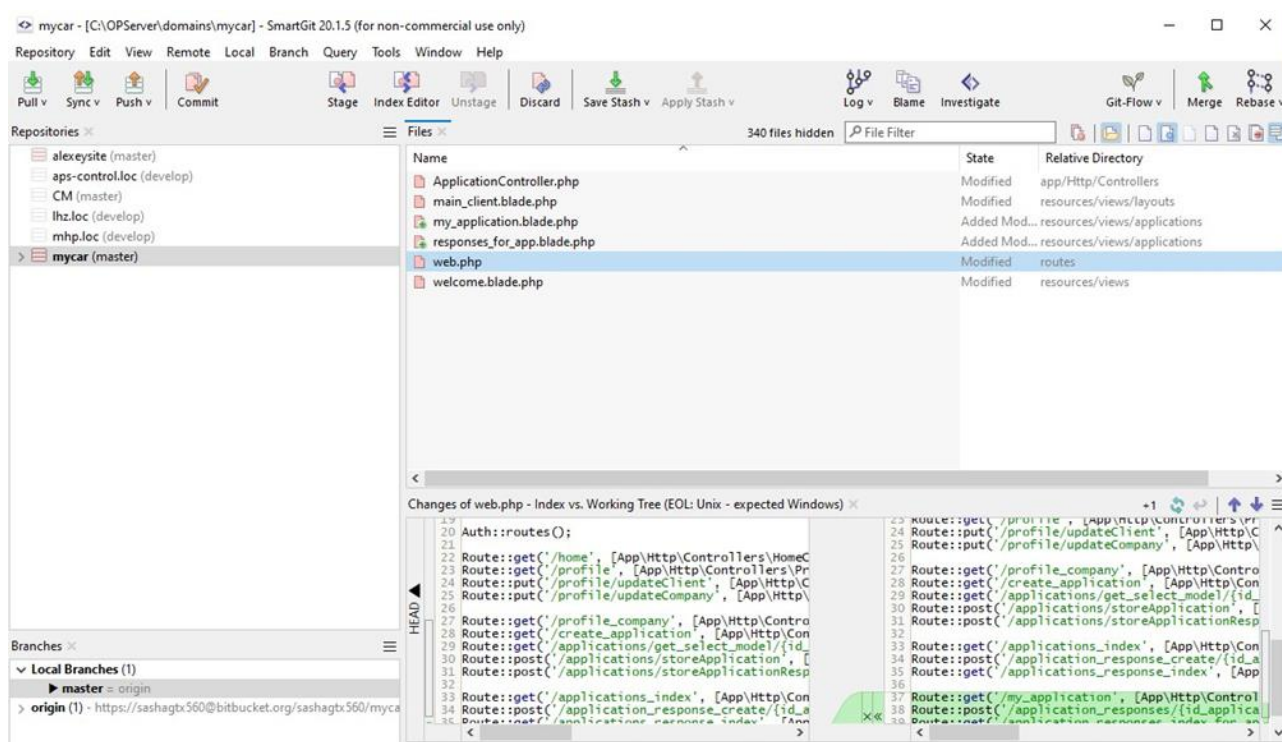


Рисунок 3.17 - Коментарі розробки

Джерело: запропоновано автором

Реалізація системи на основі Laravel розпочинається з налаштування параметрів у файлі .env. У цьому файлі зберігаються налаштування підключення до бази даних, режим розробки, налаштування електронної пошти тощо. Конфігураційний файл показано на рисунку 3.18.

Наступним кроком було створення бази даних на локальному сервері та використання вбудованого функціоналу Laravel для створення таблиць з необхідними полями та зв'язками. Цю операцію було реалізовано за допомогою файлів "міграцій". Це вбудований механізм в Laravel, який спрощує взаємодію з базою даних.

```

1 APP_NAME=MyCar
2 APP_ENV=local
3 APP_KEY=base64:ybdF4yUcwCeJw+wirxlt/WObPIe9taXog9hQBw5yIT8=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3307
12 DB_DATABASE=mycar
13 DB_USERNAME=root
14 DB_PASSWORD=root
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_DRIVER=smtp
27 MAIL_HOST=smtp.gmail.com
28 MAIL_PORT=587
29 MAIL_USERNAME=technotestlep@gmail.com
30 MAIL_PASSWORD=
31 MAIL_ENCRYPTION=tls
32 MAIL_FROM_ADDRESS=technotestlep@gmail.com
33 MAIL_FROM_NAME="${APP_NAME}"

```

Рисунок 3.18 – Конфігураційний файл .env

Джерело: запропоновано автором

Використовуючи файл міграції, показаний на рисунку 3.19, ми вказуємо поля таблиці, типи даних і необхідні зв'язки.

Потім за допомогою команди `php artisan migrate` зберігаємо конфігурацію до бази даних.

Після створення структури бази даних необхідно було додати механізм доступу до бази даних з РНР: Фреймворк Laravel використовується і працює з моделлю MVC, тому таким механізмом є модель.

Остання надає посилання на таблиці бази даних, а також показує поля, які можна редагувати.

Структура файлу моделі показана на рисунку 3.20.

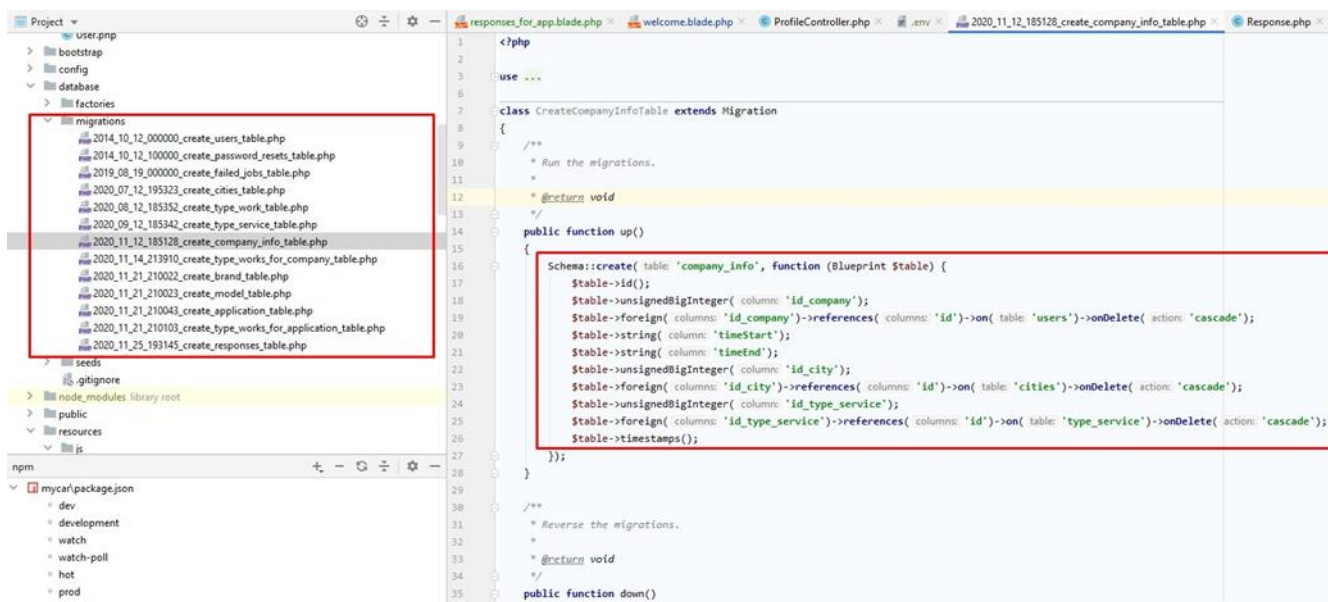


Рисунок 3.19 – Структура файлів міграцій

Джерело: запропоновано автором

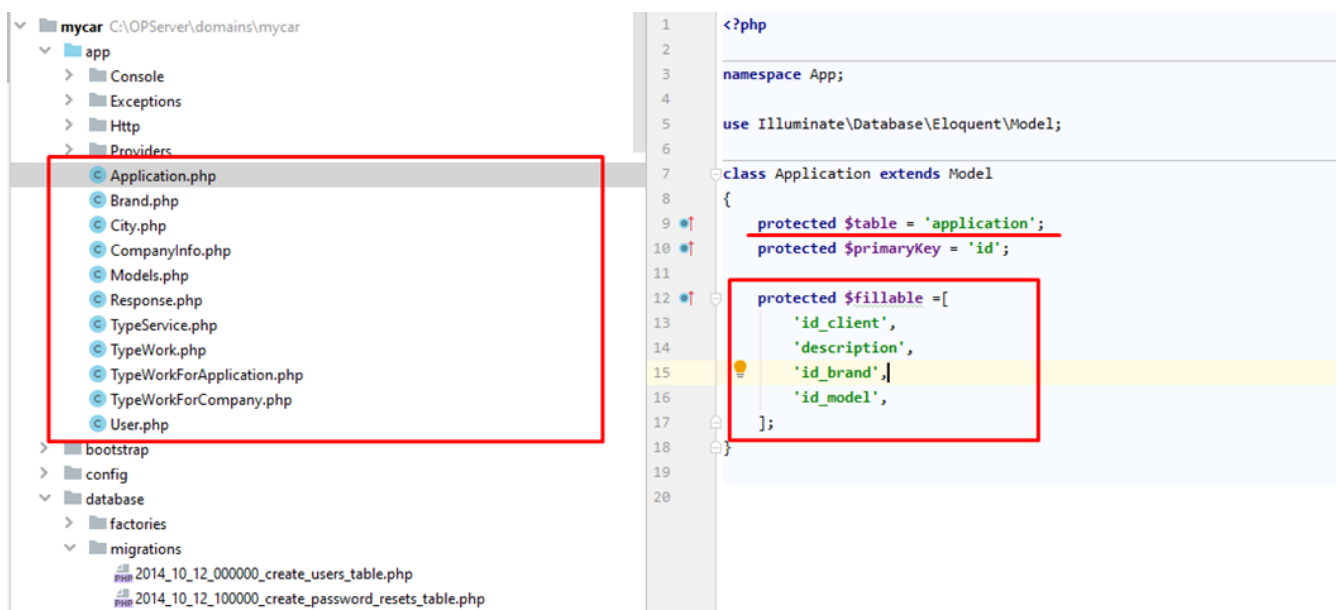


Рисунок 3.20 – Структура моделі

Джерело: запропоновано автором

Для створення логіки обробки даних необхідно визначити контролер, який збирає необхідні дані та передає їх до шаблону відображення.

Приклад контролера показано на рисунку 3.21.

Детальний програмний код наведено у Додатку Г.

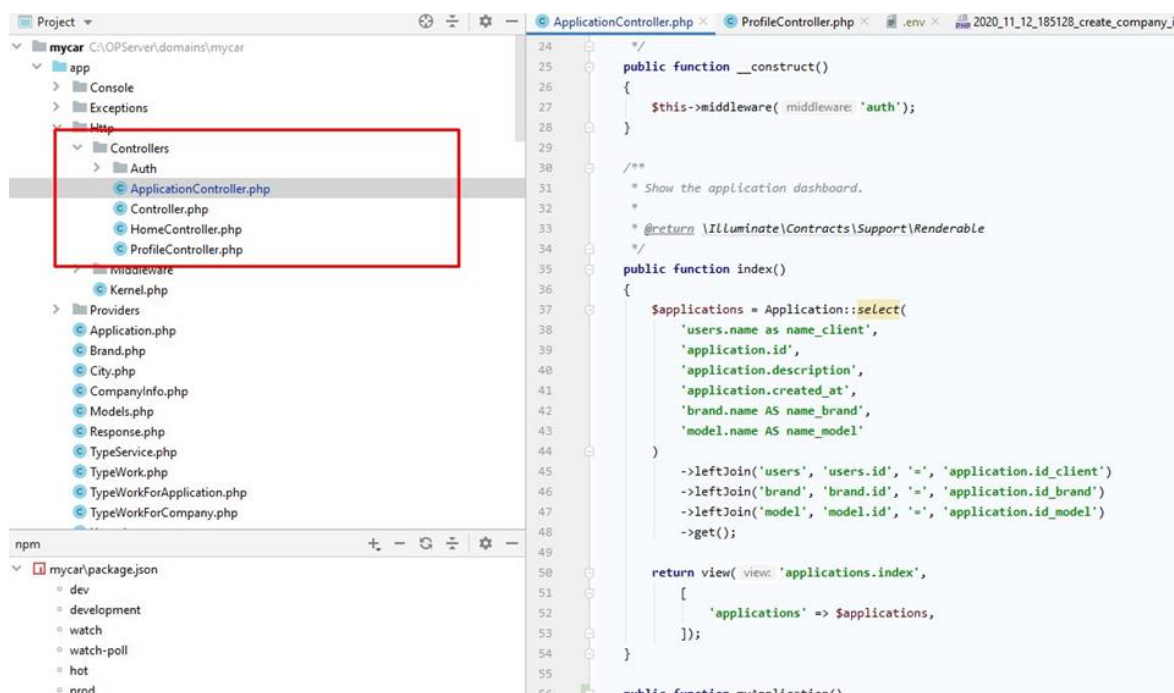


Рисунок 3.21 - Структура контролера
Джерело: запропоновано автором

Після того, як дані отримано, їх потрібно відправити до HTML-шаблонів, в Laravel використовується власний механізм blade-шаблонів. Це спрощує взаємодію між HTML та PHP. Приклад шаблону показана на рисунку 3.22, а детальний код в Додатку Г.

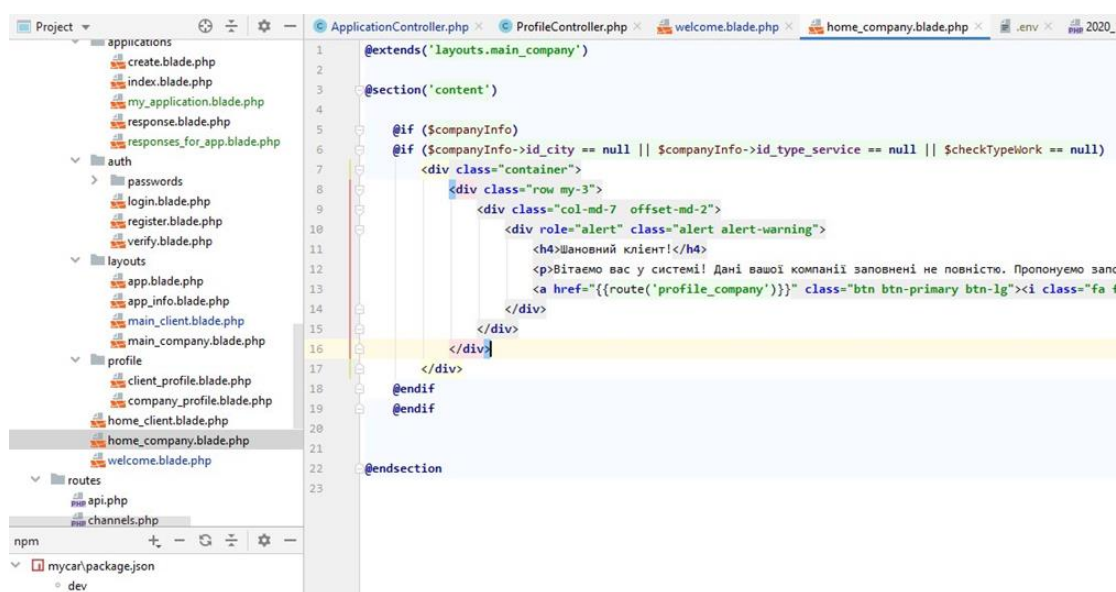


Рисунок 3.22 – Шаблон blade
Джерело: запропоновано автором

Окрім HTML та CSS, для динаміки сторінки використовується фреймворк Vue.js, Vue.js працює на основі інтерактивних компонентів. Компоненти вбудовуються в шаблон, а необхідні дані передаються від контролера до компонентів. Приклад інтеграції компонентів показано на рисунку 3.23.

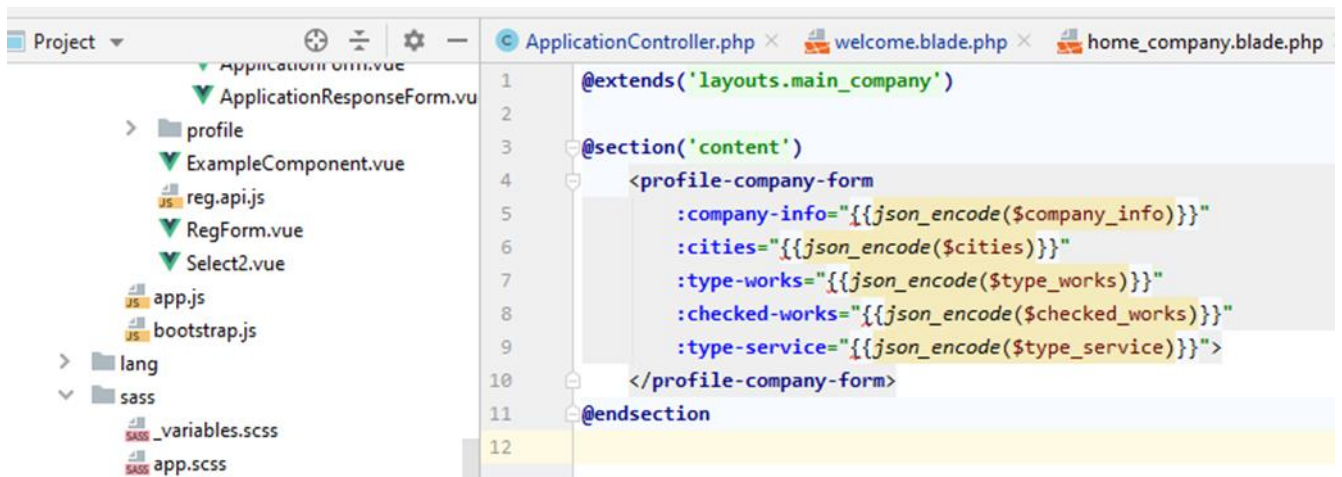


Рисунок 3.23 – Інтеграція компонентів Vue.js

Джерело: запропоновано автором

Компоненти Vue.js містять HTML-розмітку, JS-блоки для створення логіки та CSS-блоки для зміни стилів. Детальний код наведено в Додатку Г.

Вебдодаток MyCar вітає користувача на головній сторінці, як показано на рисунку 3.24.

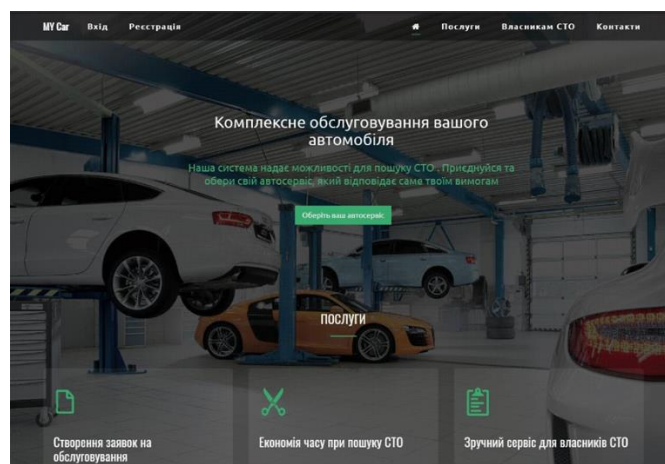
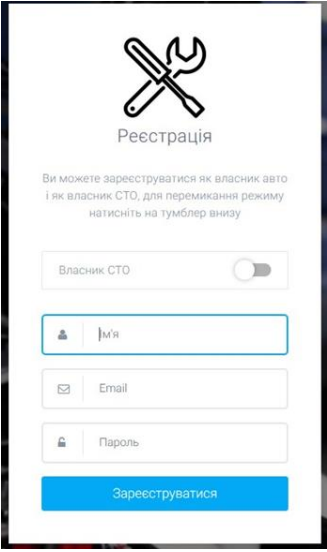


Рисунок 3.24 – Головна сторінка

Джерело: запропоновано автором

Ця сторінка містить основні пункти навігаційного меню. Основними з них є вхід та реєстрація.

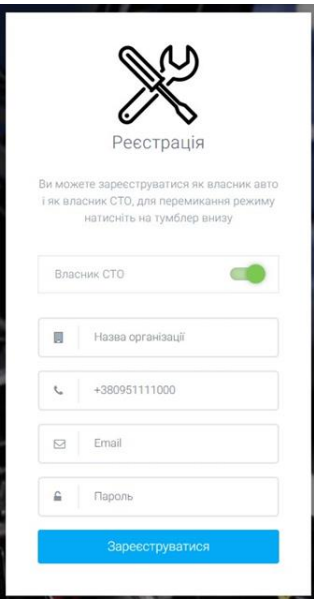
При натисканні на реєстрацію відкривається відповідна форма (рисунок 3.25). Зареєструватися може як власник транспортного засобу, так і організація. Для цього реалізовано спеціальний перемикач для динамічної зміни форми (рисунок 3.26).



The screenshot shows a mobile application interface for registration. At the top is a logo consisting of crossed wrench and screwdriver icons, with the word "Реєстрація" (Registration) below it. A text block states: "Ви можете зареєструватися як власник авто і як власник СТО, для перемикання режиму натисніть на тумблер внизу" (You can register as a car owner or as a workshop owner, to switch the mode, click the toggle switch at the bottom). Below this is a toggle switch labeled "Власник СТО" (Workshop Owner), which is currently turned off. There are three input fields: "Ім'я" (Name) with a person icon, "Email" with an envelope icon, and "Пароль" (Password) with a lock icon. At the bottom is a blue button labeled "Зареєструватися" (Register).

Рисунок 3.25 - Головна сторінка

Джерело: запропоновано автором



The screenshot shows the same registration form as Figure 3.25, but the toggle switch "Власник СТО" is now turned on. The input fields are: "Назва організації" (Organization Name) with a building icon, "+380951111000" with a phone icon, "Email" with an envelope icon, and "Пароль" (Password) with a lock icon. The blue "Зареєструватися" (Register) button remains at the bottom.

Рисунок 3.26 - Форма реєстрації клієнта

Джерело: запропоновано автором

Після реєстрації користувач може увійти в систему, форма входу в систему показана на рисунку 3.27.

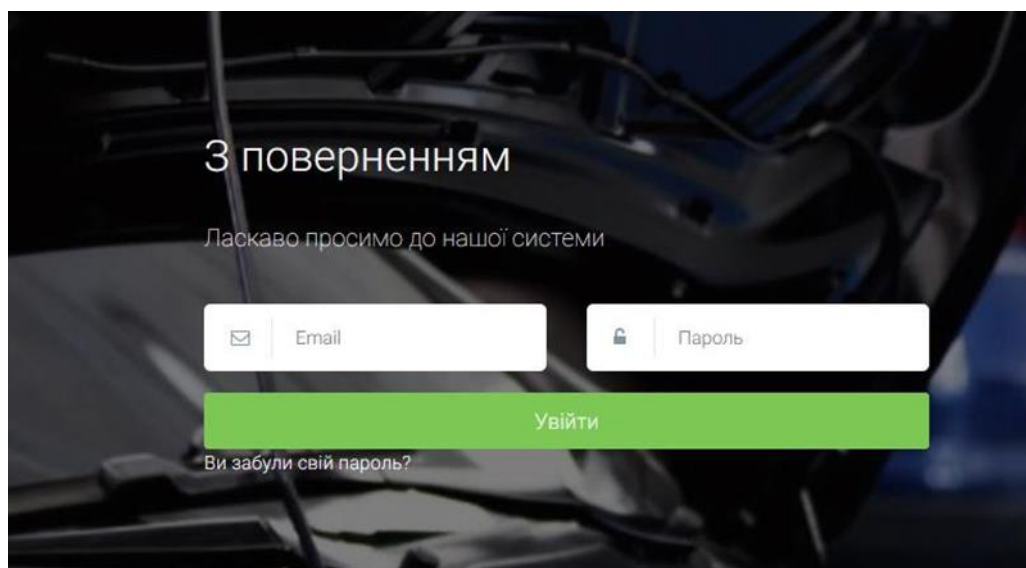


Рисунок 3.27 - Форма входу в систему

Джерело: запропоновано автором

Якщо користувач не пам'ятає свій пароль, його можна відновити за допомогою електронної пошти. Спочатку натискають на відповідне посилання у формі входу в систему, після цього відкривається форма відновлення пароля (рисунок 3.28), де користувач має ввести свою електронну адресу.

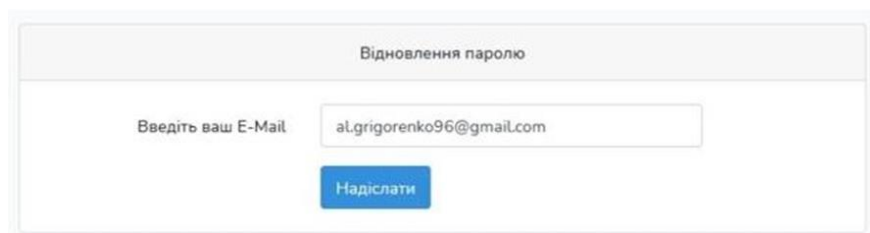


Рисунок 3.28 - Форма відновлення пароля

Джерело: запропоновано автором

Система перевірить, чи зареєстрований цей користувач. Після перевірки інформації на вказану адресу електронної пошти буде надіслано відповідний лист (рисунок 3.29) з посиланням на форму для введення нового пароля (рисунок 3.30).

Посилання в електронному листі має фіксований термін дії - 120 хвилин. Після закінчення цього часу посилання стає недійсним.

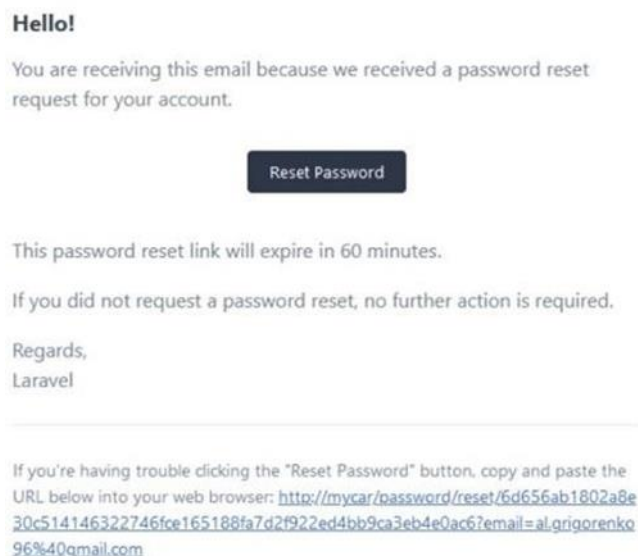


Рисунок 3.29 – Лист із посиланням
Джерело: запропоновано автором

Reset Password

E-Mail Address

Password

Confirm Password

[Reset Password](#)

Рисунок 3.30 – Форма введення нового пароля
Джерело: запропоновано автором

Після успішної аутентифікації система перевіряє тип користувача. Система має два особистих кабінети. Один для клієнтів і один для компаній.

При вході в систему відкривається один з облікових записів в залежності від типу користувача. Інтерфейси для облікового запису клієнта та компанії показані на рисунках 3.31-3.32 відповідно.

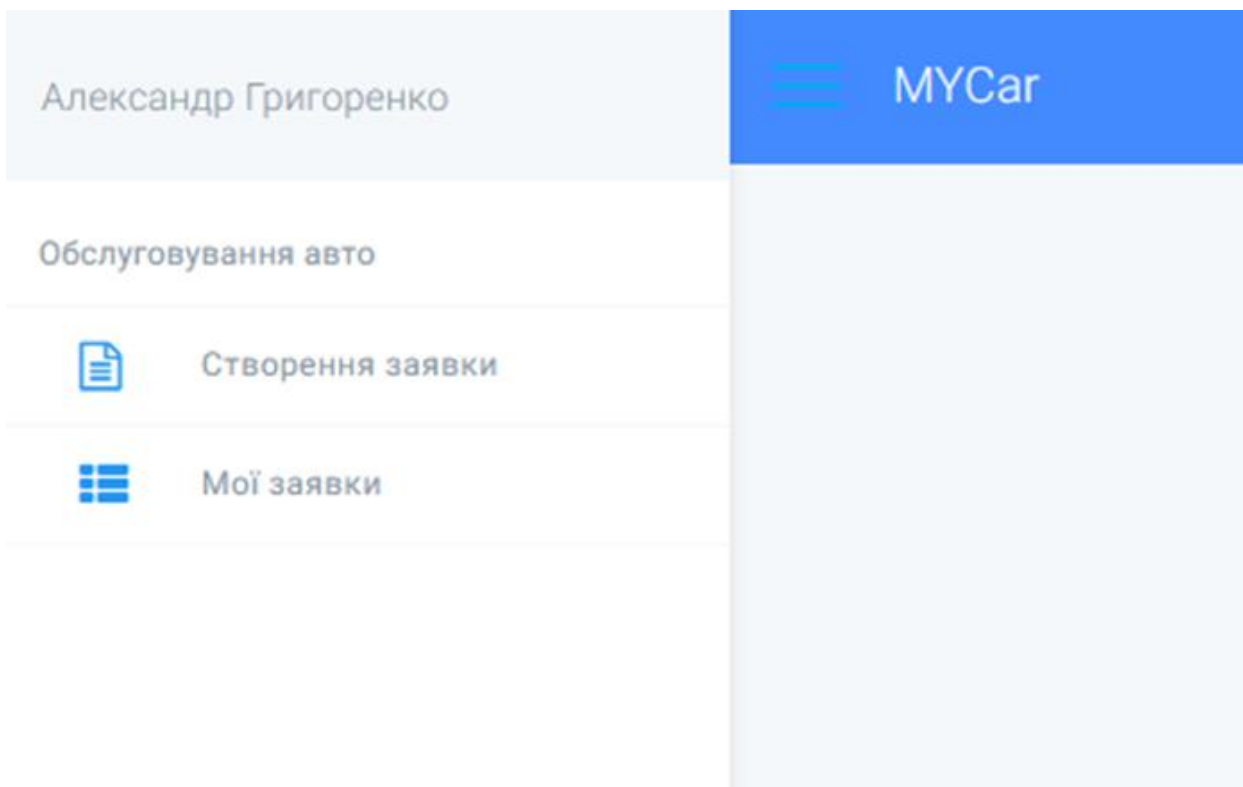


Рисунок 3.31 - Інтерфейс кабінету клієнта

Джерело: запропоновано автором

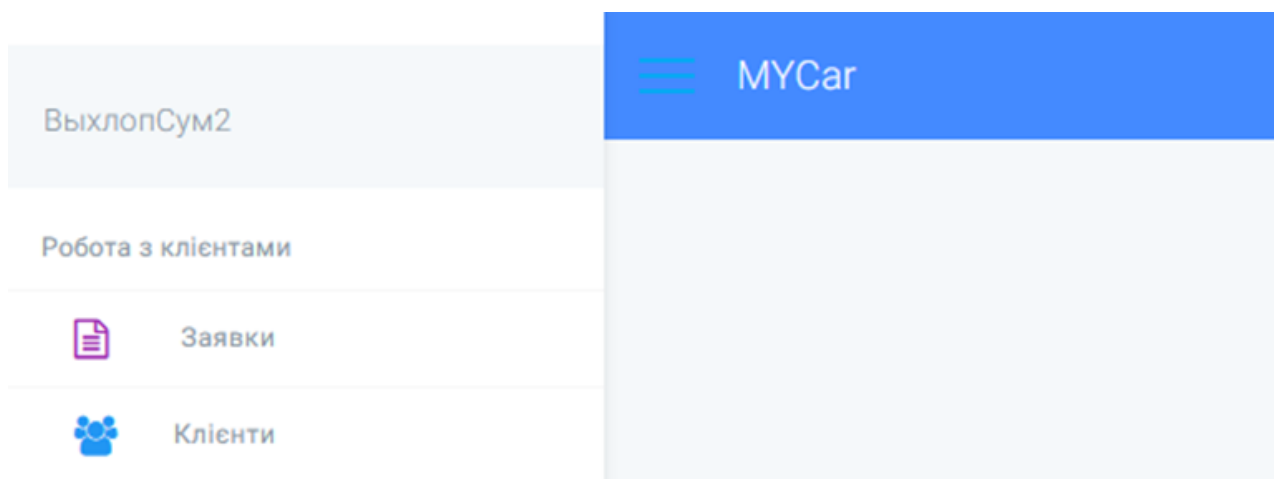


Рисунок 3.32 – Інтерфейс облікового запису компанії

Джерело: запропоновано автором

Система також дозволяє організувати персональні дані як для клієнта, так і для компанії.

Відповідні форми показані на рисунках 3.33-3.34.

Картка компанії

Назва компанії

ВыхлопСум2

✉ Email ☎ Телефон

al.grigorenko96@gmail.com ✓ +380956744001 ✓

⚙ Тип сервісу 📍 Місто

СТО Суми

☒ Сервісне технічне обслуговування
☒ Кузовний ремонт
☐ Покраска
☒ Розвал - сходження
☒ Ремонт двигуна
☒ Ремонт КПП
☒ Діагностика

 Зберегти дані

Рисунок 3.33 - Картка компанії

Джерело: запропоновано автором

Картка клієнта

Ім'я

Александр Григоренко

✉ Email ☎ Телефон

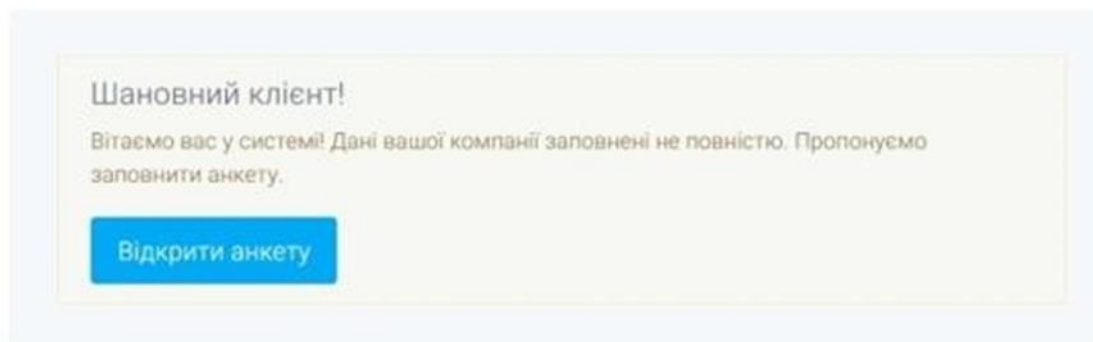
sashagtx560@gmail.com ✓ +380956744001 ✓

 Зберегти дані

Рисунок 3.34 - Картка клієнта

Джерело: запропоновано автором

У випадку з компанією здійснюються спеціальні перевірки, і якщо деякі дані незаповнені, відображається спеціальне повідомлення (рисуюнок 3.35).



Рисуюнок 3.35 – Повідомлення про недостатньо заповнені дані
Джерело: запропоновано автором

Рисуюнок 3.36 - Повідомлення про відсутність даних
Джерело: запропоновано автором

Кожен клієнт має можливість заповнити заявку на ремонт або діагностику свого автомобіля. На рисунку 3.37 показана форма створення заявки.

Рисунок 3.37 - Форма створення заявки

Джерело: запропоновано автором

Також є можливість переглянути свої запити та переглянути пропозиції від компаній, пов'язаних з ними. Інтерфейс показаний на рисунку 3.38.

Модель	Дата створення	Пропозиції
Sens	22-11-2020	
X7	22-11-2020	

Рисунок 3.38 – Перегляд запитів

Джерело: запропоновано автором

З іншого боку, компанії мають можливість переглядати запити, пов'язані з їхнім видом робіт. Якщо запит їх зацікавив, вони можуть відповісти на нього (рисунок 3.39) і згенерувати орієнтовну вартість роботи, тривалість роботи і, якщо необхідно, коментар. Форма відповіді показана на рисунку 3.40.

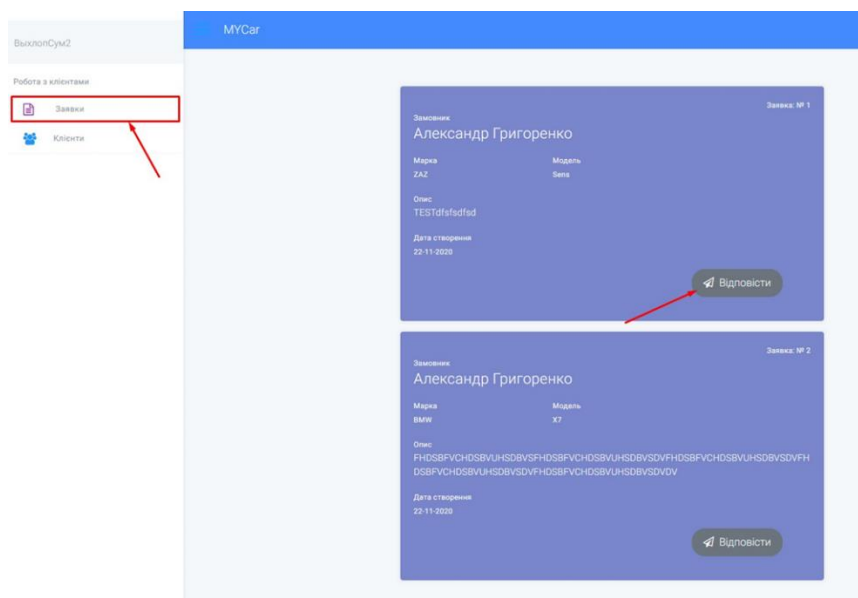


Рисунок 3.39 – Затвердження запиту компанією

Джерело: запропоновано автором

Рисунок 3.40 - Відповідь на заявку від замовника

Джерело: запропоновано автором

При відображенні пропозиції за кожним запитом система використовує внутрішній алгоритм для пошуку та надання найкращого варіанту вибору СТО.

Ця пропозиція відображається як "Вибір системи" і знаходиться вгорі (рисунок 3.41). Це дозволяє користувачу швидко вибрати найбільш вигідну пропозицію.

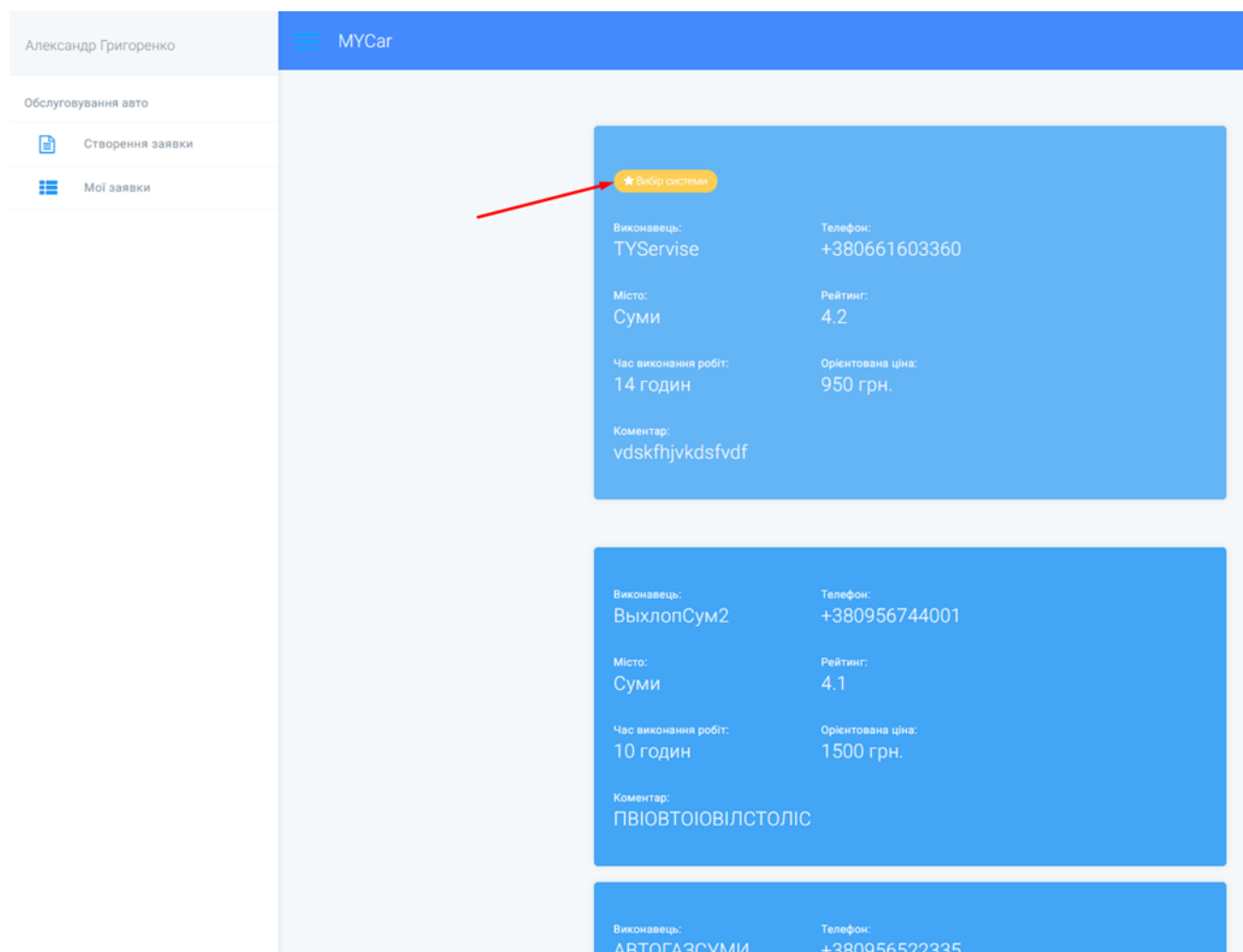


Рисунок 3.41 - Список пропозицій

Джерело: запропоновано автором

Система має панель адміністрування, де користувачі з правами адміністратора можуть переглядати та видаляти користувачів системи (рисунок 3.42).

#	Ім'я	Назва компанії	Телефон	Email		
1		ВыхлопСум2	+380956744001	al.grigorenko96@gmail.com		
3	Александр Григоренко		+380956744001	sashagtx560@gmail.com		
4		АВТОГАЗСУМИ	+380956522335	test1@gmail.com		
5		TYService	+380661603360	test2@gmail.com		

Рисунок 3.43 – Перелік користувачів

Джерело: запропоновано автором

Запропонована система дозволить автовласникам економити час та зусилля при виборі відповідної станції технічного обслуговування із запропонованих системою варіантів шляхом автоматичного пошуку найкращого рішення для обліку технічного стану легкових автомобілів відповідно до визначених вимог та критеріїв.

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці вебзастосунку для реєстрації даних моніторингу навколишнього середовища. В ході дослідження визначено вимоги до системи та проаналізовано її функціональність. Крім того, проаналізовано подібні програмні рішення та оцінено їхні сильні та слабкі сторони.

Розроблено вебзастосунок з використанням новітніх технологій, таких як HTML, CSS, JavaScript, GitHub, MVC та PHP, щоб забезпечити масштабований та ефективний продукт. Інтерфейс користувача розроблений таким чином, щоб інтуїтивно зрозумілим і простим у використанні при виконанні різних операцій, пов'язаних з природоохоронними заходами.

Розроблений вебзастосунок включає різні модулі, які виконують такі функції, як автентифікація та реєстрація користувачів, відображення та редагування даних. В рамках розробки були створені прототипи користувацького інтерфейсу сторінок сертифікації та звітності для покращення загального користувацького досвіду та полегшення взаємодії з системою.

Також була реалізована фізична база даних в MySQL для зберігання та управління інформацією, це забезпечило ефективний збір, зберігання та обробку даних, необхідних для пошуку і обліку пропозицій.

Враховуючи тенденцію розвитку автомобільного транспорту в світі, особливо в Україні, та велику кількість різноманітних СТО, особливу увагу слід приділити взаємодії між клієнтом, тобто власником автомобіля, та СТО для більш ефективної комунікації.

На ринку існує багато систем для пошуку СТО, але дуже мало з них дозволяють клієнтам знайти найбільш підходящу для них станцію технічного обслуговування, просто заповнивши кілька простих екранних форм.

Враховуючи значне збільшення кількості транспортних засобів за останні кілька років, цей ринок послуг є досить перспективним, тому була створена комплексна система для обслуговування транспортних засобів з

аналітичною підсистемою підбору СТО для пошуку найкращого рішення за конкретними критеріями користувача.

Розробка дозволить автовласникам економити час та зусилля при виборі відповідної станції технічного обслуговування із запропонованих системою варіантів шляхом автоматичного пошуку найкращого рішення відповідно до визначених вимог та критеріїв. Це також забезпечить належну організацію продуктивної взаємодії між автовласниками та станціями технічного обслуговування.

У майбутньому ця система може бути розширена додатковими модулями для задоволення більшої кількості потреб клієнтів.

В результаті роботи були досягнуті поставлені цілі та розроблено функціональний вебзастосунок, який відповідає вимогам та потребам користувачів щодо заходів з обліку технічного стану легкових автомобілів та їх обслуговування відповідно до визначених вимог та критеріїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 5 Things to Look for When Choosing a Car Service Garage URL: <https://www.dowleys.co.uk/blog/5-things-to-look-for-when-choosing-a-car-service-garage/> (дата звернення: 14.10.2023).
2. Distributorships and Dealerships URL: <https://www.inc.com/encyclopedia/distributorships-and-dealerships.html> (дата звернення: 14.10.2023).
3. Типи СТО URL: <https://vse-sto.net/kak-pravilno-vybrat-avtoservis/> (дата звернення: 14.10.2023).
4. Як обрати СТО? URL: <https://www.autocentre.ua/opyt/avtoservis/kak-vybrat-sto-5-osnovnyh-pravil-367236.html> (дата звернення: 14.10.2023).
5. Analyze insights in text with Amazon Comprehend. Amazon Web Service. URL: <https://aws.amazon.com/getting-started/hands-on/analyze-sentiment-comprehend/> (дата звернення: 14.10.2023).
6. Azure Blob Storage. Microsoft Azure. URL: <https://azure.microsoft.com/en-us/products/storage/blobs> (дата звернення: 14.10.2023).
7. Cloud Storage. Google Cloud. URL: <https://cloud.google.com/storage> (дата звернення: 14.10.2023).
8. Whitehead, T. Google Erath Blog. How Big Is the Google Earth Database? Available online: <http://www.gearthblog.com/blog/archives/2016/04/big-google-earth-database.html> (дата звернення: 14.10.2023).
9. Li, S.; Dragicevic, S.; Castro, F.A. Geospatial big data handling theory and methods: A review and research challenges. *ISPRS J. Photogramm. Remote Sens.* 2016, *115*, 119–133. (дата звернення: 14.10.2023).
10. ESA Sentinel Online. Available online: <https://sentinel.esa.int/web/sentinel/sentinel-data-access>. (дата звернення: 14.10.2023).
11. Williams, A. Financial Times. Space—The Final Frontier for Investors. Available online: <https://www.ft.com/content/05f24014-07e1-11e7-97d1->

5e720a26771b?mhq5j=e1 (дата звернення: 14.10.2023).

12. Planet. Planet launches Satellite Constellation to Image the Whole Planet Daily. Available online: <https://www.planet.com/pulse/planet-launches-satellite-constellation-to-image-the-whole-planetdaily/> (дата звернення: 14.10.2023).

13. Planet. On-demand Webinar hosted by Descartes Labs on “Turning Geospatial Data into Intelligence”. Available online: <http://info.planet.com/adp-webinar-1-on-demand/> (дата звернення: 14.10.2023).

14. Why, when, and how to use the Google Map API URL: <https://medium.com/@helennsays/why-when-and-how-to-use-the-google-map-api-f5dfa35986dc> (дата звернення: 14.10.2023).

15. Локальний сервер. URL: <https://ospanel.io/>. (дата звернення: 14.10.2023).

16. Andi Gutmans, Stig Bakken, Derick Rethans. PHP5 Power Programming. – Prentice Hall. – 704 с. (дата звернення: 14.10.2023).

17. Laravel – Функції безпеки. URL: <https://webformyself.com/laravel-funkcii-bezopasnosti/>. (дата звернення: 14.10.2023).

18. Що таке PHP? - Hyperhost Wiki. HyperHost. URL: <https://hyperhost.ua/uk/wiki/chto-takoe-php> (дата звернення: 14.10.2023).

19. Що таке CSS. URL: <http://phpist.com.ua/css/5-whatcss>. (дата звернення: 14.10.2023).

20. phpMyAdmin. phpMyAdmin. URL: <https://www.phpmyadmin.net/> (дата звернення: 14.10.2023).

21. Переваги використання Laravel в IT URL: <https://wezom.com.ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii> (дата звернення: 14.10.2023).

22. What is Laravel and Why You Should Learn it? URL: <https://www.larashout.com/what-is-laravel-and-why-you-should-learn-it>. (дата звернення: 14.10.2023).

23. What is Vue.js and How do we Use It? URL: <https://www.avantica.com/blog/what-is-vue.js-and-how-do-we-use-it> (дата звернення: 14.10.2023).

24. Vue.JS: особливості, переваги та недоліки URL:

<https://jetruby.com/ru/blog/vue-js-preimuschestva-i-nedostatki/> (дата звернення: 14.10.2023).

25. Next.js by Vercel - The React Framework. Next.js by Vercel - The React Framework. URL: <https://nextjs.org> (дата звернення: 14.10.2023).

26. Why PHP Storm? URL: <https://www.jetbrains.com/phpstorm/> (дата звернення: 14.10.2023).

27. Продуктивне використання PHPStorm URL: <https://habr.com/ru/post/157409/> (дата звернення: 14.10.2023).

28. An introduction to Git. URL: <https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/>. (дата звернення: 14.10.2023).

29. Zmerzlyi I. Архітектура веб додатків. Medium. URL: <http://surl.li/uipto> (дата звернення: 14.10.2023).

30. GitHub: Let's build from here. GitHub. URL: <https://github.com/> (дата звернення: 14.10.2023).

31. Model – view – controller URL: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> (дата звернення: 14.10.2023).

ДОДАТКИ

ДОДАТОК А

А.1 Ідентифікація мети проекту методом SMART

Визначення мети проекту проведено методом SMART на основі наступних показників: Specific (конкретна), Measurable (вимірювана), Achievable (досяжна), Relevant (реалістична), Time-framed (обмежена у часі). Результати деталізації методом SMART розміщені у таблиці А.1

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Розробити веборієнтовану систему комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО
Measurable (вимірювана)	Надати можливість для підбору оптимальної станції технічного обслуговування за зазначеними критеріями
Achievable (досяжна)	Створити веборієнтовану систему на основі реальних функціональних вимог
Relevant (реалістична)	Створити веборієнтовану систему на основі сучасних фреймворків та шаблонів дизайну
Time-framed (обмежена у часі)	Створити веборієнтовану систему у певні терміни на основі календарного плану

А.2 Планування змісту структури робіт ІТ проекту

Для планування структури спочатку необхідно визначити перелік необхідних робіт з декомпозицією великих за обсягом.

1. Фаза ініціалізації проекту.
 - 1.1. Ознайомлення з предметною областю.
 - 1.2. Первинний опис проблематики.
 - 1.3. Визначення мети проекту.
2. Фаза планування проекту.
 - 2.1. Аналіз предметної області.
 - 2.1.1. Моделювання бізнес процесів.
 - 2.1.1.1. Моделювання IDEF0.
 - 2.1.1.2. Моделювання IDEF3.
 - 2.1.2. Аналіз специфіки роботи СТО.
 - 2.2. Визначення технологій реалізації.

2.3. Аналіз вимог.

2.3.1. Визначення функціональних вимог.

2.3.2. Визначення користувацьких вимог.

2.4. Розробка календарного плану.

2.5. Визначення ресурсів та ризиків.

2.6. Визначення бюджету.

3. Фаза реалізації проекту. 3.1.Розробка дизайну.

3.1.1. Вибір задовільного дизайну.

3.1.2. Інтеграція в Laravel. 3.2.Проектування бази даних (БД).

3.2.1. Розробка концептуальної схеми.

3.2.2. Розробка логічної структури. 3.3.Розробка серверної частини.

3.3.1. Розробка алгоритму автентифікації.

3.3.2. Створення алгоритмів взаємодії з БД.

3.3.3. Розробка алгоритму створення заявок на ремонт та відповіді СТО на дані заявки.

3.3.4. Розробка алгоритму пошуку кращого варіанту на основі певних критеріїв.

3.4. Тестування.

3.4.1. Розробка контрольного прикладу.

3.4.2. Виправлення помилок.

4. Фаза завершення проекту.

4.1. Створення документації.

4.2. Захист роботи.

Ієрархічна структура робіт (Work Breakdown Structure) – це інструмент, який дозволяє розбити проект на складові частини. Вона встановлює ієрархічно структуроване розподіл робіт з реалізації проекту для всіх задіяних в ньому працівників.

Ієрархічна структура робіт представляє, по суті, перелік завдань проекту. Вона може бути представлена в графічному вигляді або у вигляді опису, який відображає вкладення робіт. Ієрархічна структура робіт організовує та визначає

весь зміст проекту. Роботи, не включені у WBS, не є роботами проекту. WBS-діаграма декомпозиції робіт наведена на рисунку А.1.

Організаційна структура проекту (Organization Breakdown Structure – OBS).

Як було зазначено вище, створення WBS здійснюється до робочого пакету, який виконується окремою групою. OBS, у свою чергу, розбивається до рівня груп, які виконують найнижчий рівень робіт у WBS. Таким чином, роботи найнижчого рівня WBS притаманні як WBS, так і OBS, тобто це фундаментальні блоки обох структур.

Якщо представити WBS по горизонтальній осі, а OBS – по вертикальній, то на перетині отримаємо елементи двоспрямованої структури, тобто певні роботи, які виконуються відповідними підрозділами проектної команди. Кожний з них має як власні ресурси, так і власний бюджет, що створює систему обліку затрат. За це відповідає менеджер-обліковець, який входить до складу адміністративної групи.

OBS-структура робіт наведена на рисунку А.2.

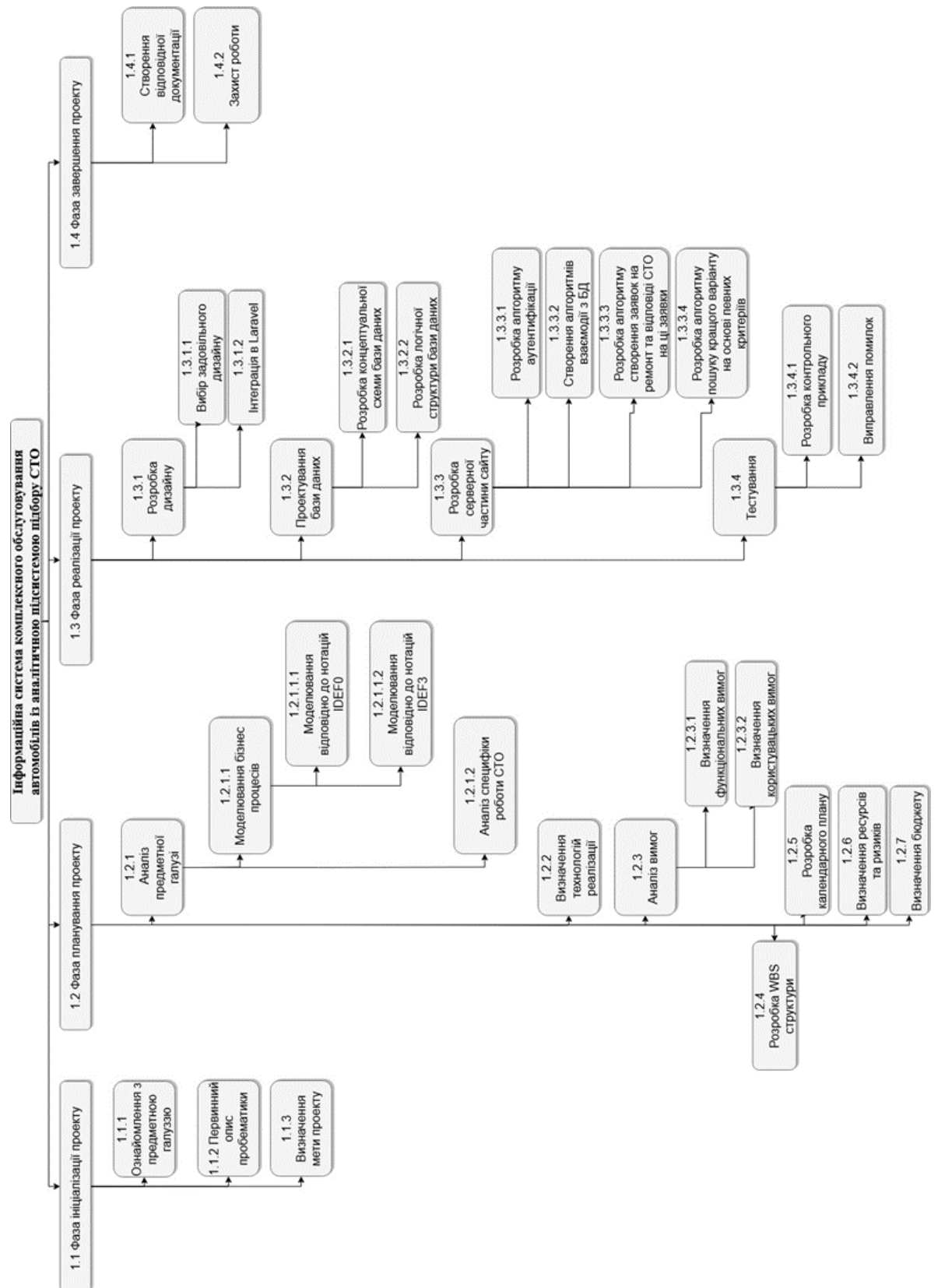


Рисунок А.1 – Діаграма декомпозиції робіт (WBS)

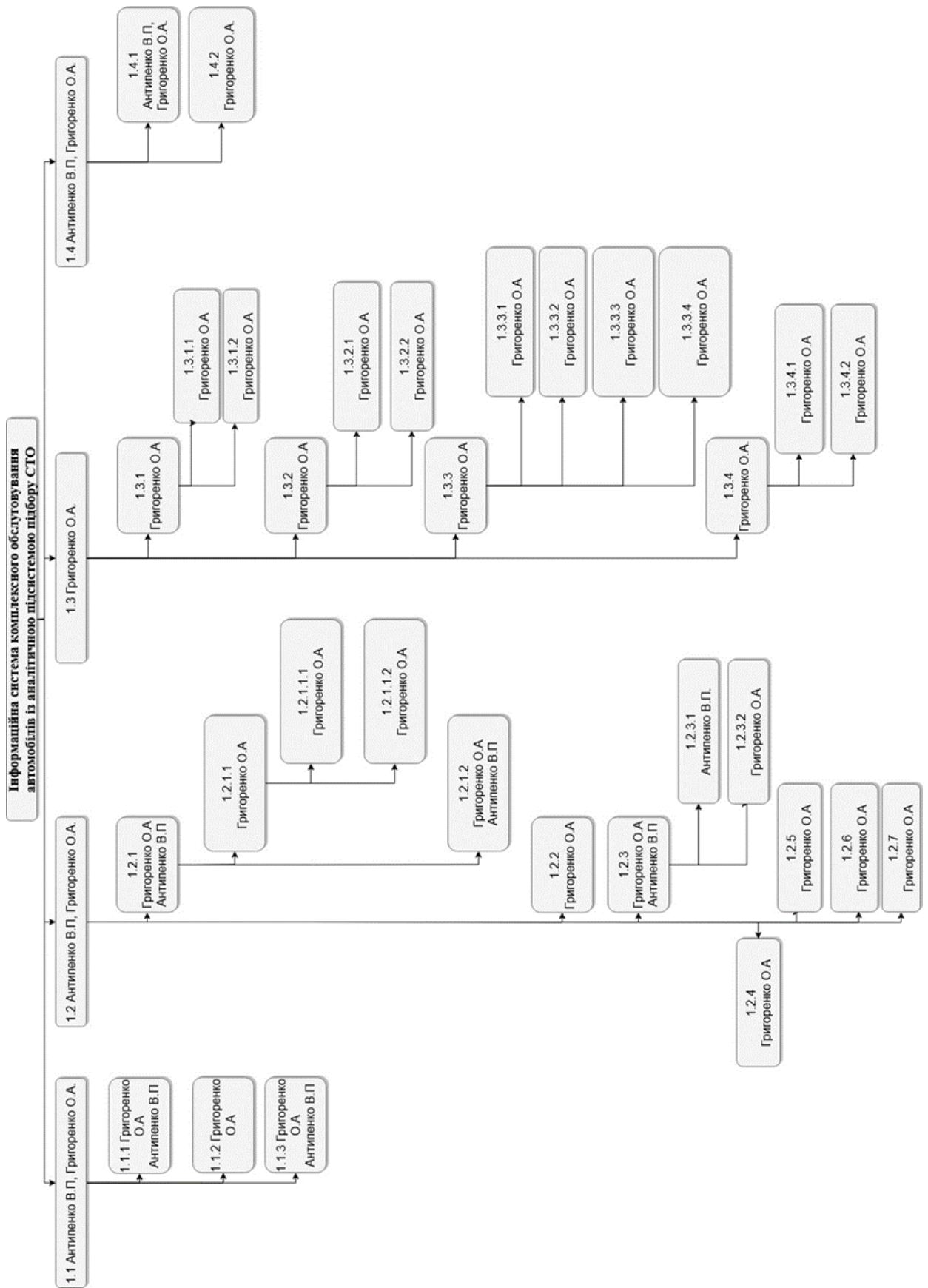


Рисунок А.2 – Діаграма організації робіт (OBS)

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

За допомогою таблиць досить незручно отримати уявлення про тривалість виконання робіт, враховувати часові обмеження.

Також потрібно брати до уваги вихідні дні при побудові графіку виконання робіт. Для даних цілей використовують візуальні інструменти для відображення робіт. Один з таких інструментів – це діаграма Ганта.

Основною частиною графіку є відображення залежності однієї виконаної роботи від іншої. Гант винайшов, що зручно формувати діаграми для кожного працівника, і результати його роботи повинні ставати основою для роботи інших працівників.

Для відображення процесу передачі завдань від однієї матеріально-відповідальної особи до іншої та необхідної кількості часу на дані дії, було створено перші діаграми Ганта, які мають вертикальну вісь, де зазначається перелік робіт, і горизонтальну – для визначення часового проміжку.

Діаграма Ганта проекту наведена на рисунках Б.1-Б.2.

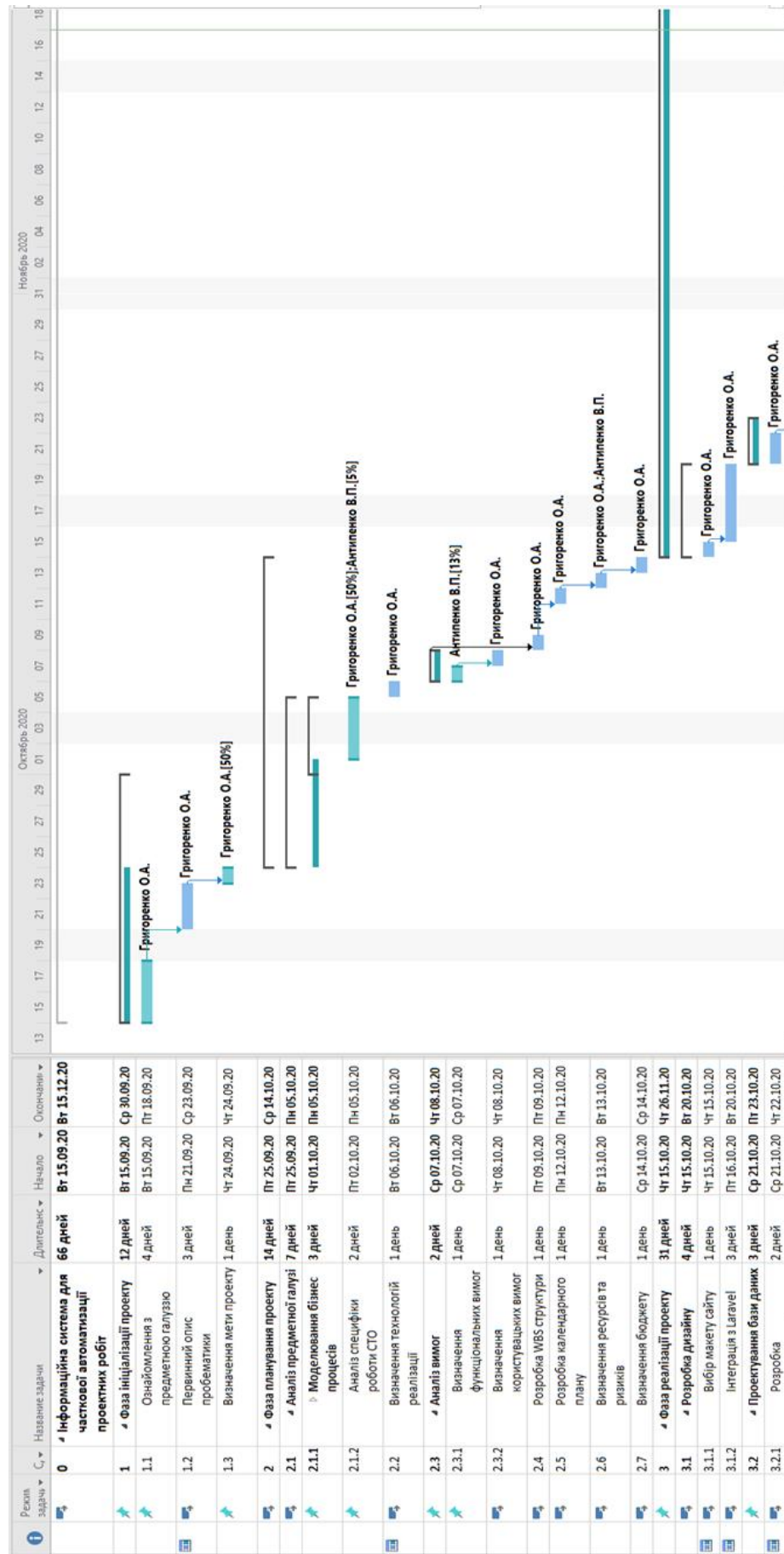


Рисунок Б.1 – Діаграма Ганта (блок 1)

Рисунок Б.2 – Діаграма Ганта (блок 2)

Планування ризиків

До основних ризиків при виконанні проекту можна віднести наступні:

- недостатньо ретельне вивчення предметної області;
- збій або відмова обладнання;
- різні погляди на реалізацію між виконавцем та замовником;
- зростання вимог;
- форс-мажорі обставини;
- зміна цілей проекту.

Ймовірність виникнення і величина ризиків даного проекту представлено в таблиці Б.1.

Таблиця Б.1 – Ймовірність виникнення і величина ризиків

№	Ризики	Виникнення	Втрати
1	Недостатньо ретельне вивчення предметної області	2	2
2	Збій або відмова обладнання	2	3
3	Різні погляди на реалізацію між виконавцем та замовником	4	4
4	Зростання вимог	3	4
5	Форс-мажорі обставини	1	3
6	Зміна цілей проекту	2	5

Далі було побудовано матрицю «Ймовірність – Втрати» для аналізу ризиків, які є найбільш небезпечні для проекту. Дана матриця наведена на рисунку Б.3.

Ймовірність	5	5	10	15	20	25
	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5
		1	2	3	4	5

Рисунок Б.3 – Матриця «Ймовірність – Втрати»

Аналізуючи дану матрицю можна розділити ризики на такі категорії:

- незначні:
 - недостатньо ретельне вивчення предметної області;
 - збій або відмова обладнання;
 - форс-мажорі обставини;
- істотні:
 - різні погляди на реалізацію між виконавцем та замовником;
- виправдані:
 - зростання вимог;
- неприпустимі:
 - зміна цілей проекту.

ДОДАТОК В
КОПІЇ ПУБЛІКАЦІЙ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**СУМСЬКИЙ НАЦІОНАЛЬНИЙ
АГРАРНИЙ УНІВЕРСИТЕТ**

МАТЕРІАЛИ

науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ

(14-16 травня 2024 р.)

Luo Hao. A STUDY ON CONSUMER BEHAVIOUR MANAGEMENT THROUGH ADVANCED MARKETING TOOLS	414
Wang Baichao. MANAGEMENT CRISIS IN AGRICULTURAL COMPANIES IN CURRENT SITUATIONS	415
Wang Ruijie. HUMAN RESOURCES MANAGEMENT OPTIMIZATION AND INNOVATION BASED ON ARTIFICIAL INTELLIGENCE	416
Yu Hang. RESEARCH ON THE INTRODUCTION OF MODERN MANAGEMENT TECHNOLOGY INTO ENTERPRISES	417
Zhou Kunming. PECULIARITIES OF MANAGING HIGH-TECH ENTERPRISES IN A TRANSPARENT ECONOMY	418
Бусенко І.В. АНАЛІЗ ОСНОВНИХ РІЗНОВИДІВ SQL-ІН'ЄКЦІЙ, АНАЛІТИКА ЗАСТОСУВАННЯ ТА СПОСОБИ ЗАХИСТУ ВІД НИХ	419
Белоконь П.А. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ПІДТРИМКИ РОБОТИ HR-МЕНЕДЖЕРА	420
Стебляк В. В. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО ПРОДУКЦІЇ. РЕАЛІЗАЦІЯ ПРОЄКТУ	421
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. ПОСТАНОВКА ЗАДАЧІ	422
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. РЕАЛІЗАЦІЯ ПРОЄКТУ	423
Биченко Є.В. РОЗРОБКА ПРОТОТИПУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ДІЯЛЬНІСТЮ ПІДПРИЄМСТВА	424
Корецький В.А. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ МОНІТОРИНГУ ІНТЕРНЕТ-ТРАФІКУ	425
Лалченко І. В. ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ	426
Хоменко Р.М. ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ЧАТ-БОТА ДЛЯ ЗРУЧНИХ ТА ЕФЕКТИВНИХ ОНЛАЙН-ЗАМОВЛЕНЬ ЕЛЕКТРОНІКИ	427
Блаженко Д. О. ПРОБЛЕМИ ПРОЄКТУВАННЯ БАЗИ ДАНИХ ПІДСИСТЕМИ ВЕДЕННЯ ЗАМОВЛЕНЬ ДЛЯ ВИГОТОВЛЕННЯ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ	428
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	429
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	430
Руденко О. А. МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ	431
Сіденко А. С. ПРОБЛЕМИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ОБЛІКУ ПРИРОДООХОРОНИ ЗАХОДІВ	432
Терещенко С.С. АВТОМАТИЗАЦІЯ ОБЛІКОВИХ ПРОЦЕСІВ: КЛЮЧ ДО ЕФЕКТИВНОГО УПРАВЛІННЯ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	433
Михайлюк О.С. ОГРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ WEB-ОРІЄНТОВАНОЇ ПІДСИСТЕМИ ПІДТРИМКИ ОБЛІКОВИХ ПРОЦЕСІВ РЕГІОНАЛЬНОГО ІНТЕРНЕТ-ПРОВАЙДЕРА	434
Олійник А.В. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО-МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	435
Чубун М.П. АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СУЧАСНОГО ВЕБ-САЙТУ	436
Дубінін В. С. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	437
Санжаков А. А. НОВИЙ ПІДХІД РОЗРОБКИ ДИЗАЙНУ ВЕБ-ЗАСТОСУНКІВ – MATERIAL 3	438
Федірко Д.Д. ПРОБЛЕМИ ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКІВ ОЦІНКИ ТАРИФНИХ ПРОПОЗИЦІЙ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	439
Федірко Д.Д. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	440
Маслак О. М. ДЕРЖАВНЕ РЕГУЛЮВАННЯ ПРОЦЕСІВ АГРОПРОМИСЛОВОГО ВИРОБНИЦТВА НА РЕГІОНАЛЬНОМУ РІВНІ (НА ПРИКЛАДІ СУМСЬКОЇ ОБЛАСТІ)	441
Горбань С.В. ІМІДЖ ПІДПРИЄМСТВА: ВПЛИВ, ФОРМУВАННЯ ТА ОЦІНКА	442
Заболотна А.І. НОРМАТИВНО-ПРАВОВЕ ЗАБЕЗПЕЧЕННЯ ЕКОНОМІЧНОГО РОЗВИТКУ ДРУЖІВСЬКОЇ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	443
Гулініна О.Ю. ШЛЯХИ ОПТИМІЗАЦІЇ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ УПРАВЛІННЯ ЗАКЛАДУ ОХОРОНИ ЗДОРОВ'Я	444
Суценко В. О. ПРОЦЕС РОЗВИТКУ ПЕРСОНАЛУ, ЯК ЧИННИК КОНКУРЕНТОЗДАТНОСТІ ПІДПРИЄМСТВА В РИНКОВИХ УМОВАХ	445
Каплун В.М. УПРАВЛІННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ ІЗ ЗАСТОСУВАННЯ ПРОГРАМНОГО КОНТРОЛЮ І МОНІТОРИНГУ	446
Бондар А.В. ТЕОРЕТИКО-МЕТОДИЧНІ ПІДХОДИ ДО ВПРОВАДЖЕННЯ ТА ОЦІНКИ СТРАТЕГІЧНОГО УПРАВЛІННЯ РОЗВИТКОМ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ	447
Романенко Т.А. СУЧАСНІ НАПРЯМИ ІНТЕНСИФІКАЦІЇ ВИРОБНИЦТВА ПРОДУКЦІЇ РОСЛИННИЦТВА В КОНТЕКСТІ ПРОДОВОЛЬНОЇ БЕЗПЕКИ	448
Мішенін Є. В., Ярова І. Є. ОРГАНІЗАЦІЙНО ЕКОНОМІЧНИЙ МЕХАНІЗМ ЗАБЕЗПЕЧЕННЯ СТАЛОГО ПРОСТОРОВНОГО РОЗВИТКУ ЛІСОВОГО ГОСПОДАРСТВА	449
Hu Guannan CRISIS MANAGEMENT THEORY	450
Zhang Liming HOW DOES BYD COMPANY IMPROVE ITS COMPETITIVENESS?	451
Zhao Jing THE INFLUENCE OF CORPORATE CULTURE ON EMPLOYEE SATISFACTION	452
Cui Xiaoping DEVELOPMENT PATH OF ENTERPRISE MANAGEMENT IN THE ERA OF ARTIFICIAL INTELLIGENCE	453

РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. ПОСТАНОВКА ЗАДАЧІ

Фідірко Т.Р., студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.т.н., доц. С.В. Агаджанова
Сумський НАУ

У сучасному світі технічний стан автомобіля відіграє важливу роль у забезпеченні безпеки, ефективності та надійності транспортного засобу. З метою покращення управління технічним обслуговуванням автомобілів та забезпечення їх надійності і довговічності виникає необхідність розробки веб-застосунку, що дозволить власникам та технічним службам здійснювати моніторинг та управління технічним станом автомобілів. Цей проект включає постановку задачі розробки такого веб-застосунку.

Метою даного проекту є розробка вебзастосунку для обліку технічного стану легкових автомобілів з можливістю моніторингу стану автомобілів, планування технічного обслуговування, аналізу витрат на ремонт та запасні частини, а також надання рекомендацій щодо оптимального обслуговування.

Задачі проекту.

1. Аналіз вимог та функціональності. Провести детальний аналіз вимог користувачів до системи.

Визначити основні функціональні можливості вебзастосунку.

2. Проектування архітектури системи. Розробити архітектуру вебзастосунку, разом з базою даних, серверною та клієнтською частинами.

Визначити структуру бази даних для зберігання інформації про автомобілі, їхній технічний стан, історію обслуговування, тощо.

3. Реалізація вебзастосунку. Розробка серверної частини застосунку з використанням вебтехнологій(можливо Node.js, Django).

Реалізувати клієнтську частину застосунку з використанням HTML, CSS, JavaScript/

Забезпечити інтеграцію з базою даних та можливість взаємодії з користувачами через інтерфейс вебзастосунку.

4. Тестування та валідація. Провести тестування розробленого вебзастосунку на різних етапах розробки.

Виконати валідацію функціональності та забезпечення високої якості програмного забезпечення.

5. Впровадження та підтримка. Впровадження розробленого вебзастосунку в експлуатацію.

Надання необхідної підтримки користувачів і забезпечення подальшого розвитку та покращення застосунку.

Очікувані результати. Очікується, що розроблений вебзастосунок забезпечить ефективне управління технічним станом легкових автомобілів, спростить процес технічного обслуговування, зменшити витрати на ремонт та запасні частини.

Проаналізовано характеристики сучасних вебплатформ з діагностики технічного стану автомобілів MyEurocar, Oiler, Oil-service, Autobooking.

Результатом проведеного дослідження є сформульовані вимоги до вебдодатку для підтримки процесу технічного обслуговування автомобілів:

Клієнтська частина.

Простий і зручний інтерфейс.

Панель користувача, для перегляду персональної інформації, інформації про автомобіль, передивитися стан виконання замовлення.

Модуль підбору СТО поблизу користувача.

Модуль для оплати онлайн за допомогою Stripe.

Модуль для аналізу характеристик транспортного засобу та підбору сервісів, що будуть корисними.

Модуль авторизації та аутентифікації.

Модуль взаємодії з мапами для відображення місця розташування СТО (Google Maps).

Модуль для генерації та розсилки сповіщень на електронну адресу.

Модуль API, що зв'язує решту модулів і формує відповіді на запити, що приходять із клієнтської частини.

Висновок. Розробка вебзастосунку для обліку технічного стану легкових автомобілів є важливим етапом у вдосконаленні управління автопарком та забезпеченні безпеки і надійності автомобілів. Цей проект спрямований на створення інноваційного і ефективного інструменту для власників авто та технічних служб з метою підвищення їхньої продуктивності та ефективності управління.

РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. РЕАЛІЗАЦІЯ ПРОЄКТУ

Фідірко Т.Р., студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.т.н., доц. С.В. Агаджанова
Сумський НАУ

Використання сучасної технології SPA для розробки веб-додатка дозволило створити Програму Figma було використано для розробки стилю вебзастосунок «ТехСервісПроєкт», вона дозволяє розробити блокову структуру(верхній, нижній, центральний блоки). На наступному ступині етапі розроблені всі окремі компоненти, визначено параметри, необхідні для опису на HTML, CSS. При цьому було створено окремий блок для вибору стилю піктограм кнопок керування, титулів, параграфів, цей блок має назву – блок зі змінними.

У вебзастосунку передбачено адаптація зображень під екрани з різною роздільною щільністю.

Всі стилі вебзастосунок представлені такою структурою: у кореневій частині розміщено загальні стилі додатку; стилі шрифтів; стилі кнопок, меню, екранних форм; календаря, сповіщення, логотипу; піктограми з інтеграцією у шрифти; стилі для визначальних елементів вебзастосунок – спливаючі вікна, модальні вікна, т.і.; окремо визначені стилі для компонентів.

Структурно вебзастосунок розбитий на наступні сторінки: головна, для реєстрації/авторизації; вибір виду робіт з обслуговування автомобіля; технічне обслуговування – майстри; діагностика; вибір необхідних для обслуговування і ремонту запасних частин, кріплення; станції технічного обслуговування; геолокація, мапа станцій технічного обслуговування; замовлення огляду автомобіля, замовлення послуг; ціни на різні види робіт з діагностики та технічного обслуговування; сторінка для оплати послуг, виконаних робіт; сторінка персональна майстрів; сторінка персональна користувачів/замовників; список автомобілів; стандартні питання/відповіді; допомога менеджера; інструкція користувача.

Навігація розроблена таким чином, що дозволяє використовувати і нижній, і верхній блоки навігації. Переходи по командам меню відбуваються зручно і просто, інтуїтивно-зрозуміло.

Головною задачею реалізації вебзастосунок було визначено зменшення ступеню впливу людського чинника, автоматизація роботи.

Так, всі дзвінки, що надходять на станцію технічного обслуговування записуються, з метою виключення помилки оператора щодо замовлення певного виду робіт, відмінити бронювання, перенести дату огляду, т.і.

Автоматизація роботи, актуалізація даних щодо цін, списку послуг, термінів виконання окремого типу замовлення, переліку запасних частин, дозволяє підвищити ефективність роботи, конкурентоспроможність окремого підприємства, оптимізувати кадровий склад, мотивувати працівників до підвищення кваліфікації. Підсистема стає більш гнучкою, діє швидше, зростає стабільність її роботи за рахунок створення резервних копій даних, що дозволяє працювати і при вимушених збоях або відмовах, і забезпечує функціональність вебзастосунок.

До основних переваг проєкту, що реалізується, «ТехСервісПроєкт» слід віднести наступні:

- Надання можливості вибору пунктів для формування замовлення за рахунок зрозумілості та наявності сценаріїв;
- Надання можливості вибору виду сервісу;
- Надання можливості замовнику самому обирати необхідні види техобслуговування до пункту Оплати, з опцією відміни будь-якого із обраних пунктів і повернення до головного меню;
- Проведення розрахунку вартості послуг та оплата в онлайн(при обов'язковій реєстрації);
- Відсутність «прихованих» оплат чи цін на послуги, все виконане прозоро;
- Можливість формування договору за послуги;
- Забезпечення свободи вибору надавачів послуг за рахунок доступу до мапи станцій технічного обслуговування;
- Забезпечення зворотнього зв'язку клієнт-менеджер, клієнт-майстер;
- Конфіденційність проведення платежів, шифрування даних;
- Простий й зрозумілий інтерфейс користувача.

Висновок. Вебзастосунок «ТехСервісПроєкт» розроблений за всіма критеріями та вимогами, що було визначено на етапі постановки задачі, аналізу існуючих програмних рішень, що використовуються для організації роботи автоматизованих інформаційних додатків на станціях та у центрах технічного обслуговування(сервісу) України, має розвинену функціональність, передбачає можливість подальшого вдосконалення та розвитку.

ДОДАТОК Г

ЛІСТИНГИ ПРОГРАМ

Г.1 HomeController

```
<?php

namespace

App\Http\Controllers;use

App\CompanyInfo;
use
App\TypeWorkForCompany;
use App\User;
use Illuminate\Http\Request;

class HomeController extends Controller
{
/**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

/**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        $user_type = \Auth::user();

        if($user_type->isCompany == 1){

            $companyInfo = CompanyInfo::where('id_company', $user_type->id)-
            >first();

            $checkTypeWork = TypeWorkForCompany::where('id_company', $user_type-
            >id)->first();

            return view('home_company', [ 'companyInfo' =>
            $companyInfo, 'checkTypeWork' =>
            $checkTypeWork,
            ]);
        }
        elseif ($user_type->isCompany == 0){
            return view('home_client');
        }
    }

    public function indexAdmin()
    {
        $user_type = \Auth::user();

        if($user_type->isAdmin == 1){
```

```

$users = User::all();

return view('admin_index', [

    'users' => $users,
]);
}
else{
return view('welcome');
}
}

}

```

Γ.2 ProfileController

```

<?php

namespace

App\Http\Controllers;use

App\City;
use
App\CompanyInfo;
use
App\TypeService;
use App\TypeWork;
use
App\TypeWorkForCompany;
use App\User;
use Illuminate\Http\Request;

class ProfileController extends Controller
{
/**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

/**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        $user_info = \Auth::user();

        if($user_info->isCompany == 1){ return
        view('company_profile');
        }
        elseif ($user_info->isCompany == 0){
        return view('profile.client_profile', compact('user_info'));
        }
    }

    public function updateClientProfile(Request $request)

```

```

{
$user = \Auth::user();

$user->name = $request->name;
$user->email = $request->email;
$user->phone = $request->phone;

$user->save();

return true;
}

public function updateCompanyProfile(Request $request)
{
$company = \Auth::user();

$company_info = CompanyInfo::where('id_company',$company->id)->first();

$company->name_company = $request->name_company;
$company->email = $request->email;
$company->phone = $request->phone;

$company->save();

if($company_info){
$company_info->id_city = $request->city;
$company_info->id_type_service = $request->type_service;

$company_info->save();
}
else{
$new_company_info = new CompanyInfo([
'id_company' => $company->id, 'id_city' =>
$request->city,
'id_type_service' => $request->type_service,
]);

$new_company_info->save();
}

TypeWorkForCompany::where('id_company', \Auth::user()->id)->delete();

foreach ($request->checked_works as $key => $value) {
$type_work = new TypeWorkForCompany([
'id_company' => $company->id, 'id_type_work'
=> $value,
]);
$type_work->save();
}

return true;
}

public function profileCompany()
{
$company_info = User::select(
'users.name_company',
'users.phone', 'users.email',
'company_info.timeStart as timeStart',
'company_info.timeEnd as timeEnd', 'cities.id
as city',

```

```

'type_service.id as type_service',
)
->leftJoin('company_info', 'company_info.id_company', '=', 'users.id')
->leftJoin('cities', 'cities.id', '=', 'company_info.id_city')
->leftJoin('type_service', 'type_service.id', '=',
'company_info.id_type_service')
->where('users.id', \Auth::user()->id)
->where('users.isCompany', 1)
->first();

return view('profile.company_profile', [
'company_info' => $company_info,
'cities' => City::select('id', 'name')->get()->toArray(), 'type_service' =>
TypeService::select('id', 'name')->get()->toArray(), 'type_works' =>
TypeWork::select('id', 'name')->get()->toArray(), 'checked_works' =>
TypeWorkForCompany::select('id_type_work')-
>where('id_company', \Auth::user()->id)->get()->toArray(),
]);

}

}

```