МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Веб-орієнтована інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-91/2 Григоренко Олександр Андрійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2020 р.

Науковий керівник _____ к.т.н. Антипенко В.П.
(підпис)

Голова комісії _____ Шифрін Д.М
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2020 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Григоренко Олександр Андрійович
(прізвище, ім'я, по батькові)

1 Тема проекту *Веб-орієнтована інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування*

затверджена наказом по університету від «___»_____2020 р. № _____

2 Термін здачі студентом закінченого проекту «_07__» ___грудня___ 2020 р.

3 Вхідні дані до проекту Необхідність створення інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Аналіз предметної області, постановка задач досліджень, проектування веб-орієнтованої інформаційної системи, процес розробки інформаційної системи

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Актуальність, мета та задачі роботи, вимоги до системи, моделювання роботи системи, діаграма варіантів використання, структура бази даних, використані технології та засоби, схема пошуку оптимальної пропозиції, демонстрація роботи, акт впровадження та тези ІМА2021, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Отримання завдання	15.09.20 – 16.09.20	
2	Аналіз предметної області	16.09.20 – 18.09.20	
3	Опис проблематики	21.09.20 – 23.09.20	
4	Планування проекту	25.09.20 – 14.10.20	
5	Розробка дизайну	15.10.20 – 20.10.20	
6	Проектування бази даних	21.10.20 – 23.10.20	
7	Розробка серверної частини	26.10.20 – 23.11.20	
8	Тестування	24.11.20 – 26.11.20	
9	Створення документації	27.11.20 – 09.12.20	
10	Захист роботи	10.12.20 – 21.12.20	

Магістрант _____ Григоренко О.А.

Керівник роботи _____ к.т.н. Антипенко В.П.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Веб-орієнтована інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування».

Пояснювальна записка складається з 4 розділів, вступу та висновків, списку використаної літератури та 3 додатків. Загальний обсяг роботи − 85 сторінок.

Кваліфікаційну роботу магістра присвячено розробці веб-орієнтованої інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування для спрощення взаємодії між власниками авто та компаніями.

У роботі проведено аналіз предметної області та останніх досліджень, здійснено постановку задач, визначено засоби реалізації й виконано планування та проектування робіт.

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

Результатом роботи є веб-орієнтована інформаційна система комплексного обслуговування автомобілів.

Практичне значення даної роботи полягає у тому, що розроблена інформаційна система дозволить автовласнику скоротити час та зусилля щодо вибору потрібної СТО з представлених системою варіантів за рахунок автоматизованого пошуку оптимального рішення на основі певних вимог та критеріїв заданих користувачем. Також це забезпечить належну організацію продуктивної взаємодії автовласників та станцій технічного обслуговування.

Ключові слова: сайт, СТО, інформаційна система, авто, автовласник, ремонт.

ЗМІСТ

ВСТУП

Сучасне життя кожної людини невід'ємно пов'язане з регулярними перевезеннями, в особливості з застосуванням автомобільного транспорту. Останній в деяких країнах є більш доступним для населення, а в інших – менше. Але сьогодні автомобілі є не тільки засобом пересування, але й одним із основних компонентів, на яких тримається економіка держав. Вантажні перевезення, транспортування населення, збір врожаю тощо – усе це неможливо без використання автотранспорту.

Зараз має місце тенденція, що з кожним роком кількість авто значно збільшується. Статистика показує її стрімке зростання за останні п'ять років в Україні. Необхідно зазначити, що ера бензинових двигунів поступово відступає. Їм на заміну приходять електрокари. Але можливості останніх та їх висока ціна ще не можуть цілком замінити бензинових аналогів. Однак прогрес за останні роки є досить суттєвим.

Також спочатку до нашої країни ввозили автомобілі з Європи. Але вже сьогодні ринок заполонили привезені авто із США. Це зумовлено їх високими цінами на внутрішньому ринку, що й спонукає людей шукати нові можливості стати автовласником без переплати.

Актуальність. Окрім придбання автомобіля також необхідно задуматися про його своєчасне обслуговування. Багато людей нехтують даним питанням і продовжують їздити на авто, яке потребує належної перевірки та/або апгрейду, що є досить небезпечним. Цьому сприяє ряд причин, наприклад, менталітет нашої країни, невідповідальність і неуважність людини, низький рівень фінансових доходів тощо. Заміна мастил, охолоджувальної рідини, шин із літніх на зимові та інше – це все те, що так чи інакше потрібно робити автовласнику. Раніше автомобілів було мало, а станцій технічного обслуговування (СТО) – ще менше. Дефіцит тих чи інших запчастин змушували автовласників займатися обслуговуванням власного транспорту власноруч. Із прогресом ситуація змінилася

й на сьогодні такої проблеми немає. СТО стали прибутковим бізнесом, а, отже, й легкодоступними майже у кожному куточку світу завдяки власному розповсюдженню. Через безліч варіантів власнику авто іноді навіть складно обрати для себе оптимальний сервіс по ціні, якості, місцезнаходженню тощо.

Дійсно, у мережі Інтернет є багато сайтів, на яких можна знайти ту чи іншу інформацію про наявні станції технічного обслуговування. Багато з них повідомляють про власні переваги, а деякі навіть надають можливість пошуку СТО за типом робіт. Але це практично не допомагає власнику авто у виборі. Переглянувши з десяток таких сайтів, які є схожими між собою, людині важко прийняти рішення на користь одного з них, оскільки всім їм бракує певного функціоналу для пошуку оптимального варіанту за обраними критеріями користувача. У свою чергу, це зменшить час та зусилля автовласника для визначення найбільш відповідної його можливостям і вподобанням СТО.

Тому, **метою даної роботи** є розробка інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО для пошуку оптимального рішення за певними критеріями користувача.

Об’єкт дослідження. Процеси надання комплексних послуг із технічного обслуговування автомобілів та вибору СТО.

Предмет дослідження. Інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО.

Практичне значення даної роботи полягає у тому, що розроблена інформаційна система дозволить автовласнику скоротити час та зусилля щодо вибору потрібної СТО з представлених системою варіантів за рахунок автоматизованого пошуку оптимального рішення на основі певних вимог та критеріїв заданих користувачем. Також це забезпечить належну організацію продуктивної взаємодії автовласників та станцій технічного обслуговування.

1 АНАЛІЗ ПРЕДМЕТИНОЇ ОБЛАСТІ

1.1 Аналіз процесу вибору станцій технічного обслуговування автомобілів

У сучасному світі роль автомобілів неможливо недооцінити. Важко уявити, що такого стрімкого прогресу людство досягло протягом останніх ста років.

Автомобіль – це досить складний механізм, до якого входить велика кількість систем та пристроїв.

Згодом автовиробники проводять поліпшення моделей, але, незважаючи на якість і міцність, із часом будь-які механізми можуть вийти з ладу.

Сучасні автомобілі досить технологічні. Тому для діагностики несправного механізму або системи необхідне спеціалізоване обладнання, яке є тільки на станціях технічного обслуговування.

Кожен автовласник прагне віддати власне авто в руки професіоналів для проведення ними належного обслуговування.

Вибираючи автосервіс, потрібно звернути увагу на наступні аспекти:

– перелік послуг, які надаються;
– наявність складу або магазину запчастин;
– якість наданих послуг та дотримання термінів;
– існування додаткових послуг.

Варто зазначити, що чудовий автосервіс не приховує інформацію від власних клієнтів і веде бізнес відкрито.

Також необхідно не ігнорувати додаткові аспекти [1] при виборі, а саме:

– досвід (якщо компанія давно на ринку, то вірогідність того, що там працюють професійні фахівці з досвідом вище);
– гарні рекомендації (це можуть бути як поради знайомих або відгуки на спеціалізованих ресурсах);
– справедливі ціни (вартість робіт не повинна бути дуже низькою, але в

той же час занадто високою).

На даний момент є декілька типів сервісів для автомобілів, які відрізняються розмірами, видами та якістю послуг. Вони є наступними:

– **офіційні дилери** – це великі технічні центри, які безпосередньо співпрацюють із автовиробниками. Автосалон іноді називають роздрібним дистриб'ютором. На відміну від інших типів франчайзі, включаючи деяких дистриб'юторів, дилер рідко пропонує одну лінійку товарів [2]. У роботі такі сервіси використовують лише оригінальні запасні частини. Їх працівники мають досвід і навички, щоб впоратися з будь-яким видом ремонту. Однак у таких станцій є недолік, а саме висока вартість послуг;

– **незалежні сервіси** бувають спеціалізованими (експерти працюють із невеликою кількістю марок авто) і універсальними (співробітники мають справу з будь-якими марками машин і мають великий досвід роботи). Вартість послуг в рамках таких СТО набагато нижче, ніж у офіційних;

– **приватні майстерні** – це такі сервіси, які часто розташовуються в гаражі. Іноді ремонт у них є ризикованою справою. Вони відрізняються досить низькими цінами. Однак і серед таких майстрів є відмінні фахівці з величезним досвідом [3].

Також автовласникам варто звернути увагу на декілька факторів [4], які визначають сумнівну станцію технічного обслуговування, а саме:

– **нечемні співробітники**: на гідному підприємстві не будуть тримати працівника, який грубо поводиться з клієнтами;

– **нав'язування придбання запчастин**: є нечемним, якщо споживачам нав'язують комплектуючі або тільки витратні оригінальні матеріали або матеріали однієї марки. Обмежуючи їх свободу вибору, СТО намагається заробити на цьому більшу кількість грошей і уникнути зайвого клопоту. При цьому ігноруються побажання та/або можливості автовласників;

– **умови роботи для співробітників майстерні**: необхідно проаналізувати умови роботи та звернути увагу на мікроклімат у приміщеннях, оскільки це може надати інформацію щодо майбутньої якості обслуговування;

– **обмеження**: є такі майстерні, де немає кімнати відпочинку з видом на цех і відсутня можливість перебування клієнта в зоні ремонту за будь-яких умов. Не варто довіряти власну машину таким фахівцям, які будь-що приховують від автовласника;

– **нав'язування послуг, а не рекомендації:** при діагностиці несправності автомобіля не кожен клієнт може зрозуміти різницю між рекомендацією спеціаліста й бажанням нав'язати йому більше товарів або послуг.

У результаті проведеного аналізу, можна зробити висновок, що факторів при підборі станції технічного обслуговування досить багато. Усі вони ускладнюють вибір, особливо, якщо доводиться здійснювати пошук інформації вручну по кожному СТО окремо.

1.2 Аналіз наявних послуг на станціях технічного обслуговування

Сервісні та ремонтні майстерні по типам робіт можна розділити на дві основні категорії. Перша – це станції комплексного обслуговування, які надають всі види послуг. Друга – спеціалізовані майстерні. Вони надають лише певний тип обслуговування.

Основні види робіт на станціях технічного обслуговування є наступними:

– регламентне технічне обслуговування;
– кузовний ремонт;
– обслуговування або ремонт двигуна;
– обслуговування або ремонт коробок передач;
– ремонт ходової частини;
– розвал сходження;
– ремонт електрики;
– обслуговування газобалонного обладнання;
– діагностика автомобіля.

Також можуть надаватися такі додаткові послуги:

– обслуговування або ремонт кондиціонерів;
– шиномонтажні роботи;
– встановлення мультимедійних систем;
– тюнинг та стайлінг авто.

Як видно з вище зазначеного, різноманітність наявних сервісних послуг по обслуговуванню автомобілів є досить широкою. А, отже, є потреба в розробці такої інформаційної системи, яка б по введеним критеріям користувача, в першу чергу типам робіт, виконувала пошук потрібної СТО, надаючи оптимальний вибір серед безлічі існуючих. Це скоротить час і зусилля клієнта на підбір такої майстерні, яка відповідатиме його можливостям і вподобанням.

1.3 Перспективи використання web-технологій для розробки інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО

На сьогоднішній день багато аспектів сучасного життя переноситься в інформаційну мережу, прискорюючи розвиток суспільства та долаючи географічні перешкоди. Інтернет надає широкі можливості для комунікації людства через web-сторінки – це сукупність програмних, медійних та інформаційних засобів, які логічно пов’язані між собою.

У сучасних умовах ведення бізнесу майже кожна організація має власне представництво в мережі Інтернет. Це є віддзеркаленням її успішності та конкурентоспроможності. Переваги використання web-сайтів є наступними:

– **веб-сторінки встановити дешевше та простіше**: завдяки сайтам компанії мають можливість знизити витрати. Це стосується саме відділу ІТ, який займається встановленням ПО та його подальшою підтримкою. У цьому ж випадку

користувач має мати лише персональний комп’ютер або смартфон із виходом у Інтернет;

– **оновлення є дешевшим і менш складним**: Окрім розробки велике значення має вартість обслуговування програмного забезпечення (ПЗ). Регулярне оновлення – це запорука тривалому життю для програмного продукту. Зважаючи на це можна відзначити, що переваги, які були згадані раніше, можна застосувати до даної ситуації. Додаток оновлюється тільки на сервері, в результаті всі клієнти відразу матимуть можливість працювати з новою версією програмного продукту;

– **більш універсальні та практичні**: зацікавленій особі достатньо встановити web-додаток на сервер, який працює під будь-якою сучасною операційною системою (ОС). Тоді клієнти матимуть до нього доступ через мережу Інтернет з різних пристроїв, які працюють під управлінням Mac OS, Windows, Android або іншої ОС. Якщо web-додатки розроблені якісно з використанням сучасних технологій розробки, вони будуть функціонувати однаково добре в будь-якому браузері: Mozilla Firefox, Opera, Google Chrome або Safari;

– **полегшують процес зберігання даних**: якщо існує необхідність мати доступ до однакових даних із різних точок, тоді простіше організувати їх зберігання в одному місці. В результаті відпадає необхідність синхронізації.

Для забезпечення актуальності web-сайту серед користувачів, потрібно організувати постійний зворотний зв’язок, наприклад, через електронну пошту, чати, форуми та ін.. – усі інструменти, через які гості ресурсу можуть висловити власні прохання, вподобання, рекомендації тощо.

Також, у наш час інтерактивний web-сайт має можливість підтримувати зв’язок між адміністрацією та відвідувачами в режимі реального часу, без будь-яких додаткових коштів. Для цього існують такі системи як онлайн-консультація, онлайн-чат (системи, які миттєвого проводять обмін повідомленнями прямо на web-сайті) й інші подібні інструменти.

Інтерактивний web-сайт для бізнесу дозволяє його власникам отримати додатковий дохід, завдяки активному залученню клієнтів, а відвідувачам – мати

можливість користуватися зручним і якісним сервісом, який без сумнівів, вигідно виділить відповідну організацію на тлі її конкурентів [5].

Тенденції web-відображення даних. У перші ери web-картографування геопростору дані та карти створювались професіоналами (наприклад, зйомка, фотограмметрія та дистанційне зондування) і мали невеликі розміри.

Згідно з Білою книгою Cisco [6], кількість даних, доступних в мережі Інтернет, зросла до більш, ніж одного мільярда web-сайтів до кінця 2016 року [7] з обсягом трафіку більше 1 зеттабайт (1000 ексабайт) на рік. Очікується, що обсяг трафіку збільшиться до 2,3 ZB на рік до 2021. Це включає тенденцію переміщення даних із середовища робочого столу в хмарне через збільшення обсягу даних. Останні збираються за допомогою мобільних пристроїв, датчиків та соціальних мереж спілкування платформи.

Із цих даних велику частку становлять геопросторові зображення та об’єкти. Наприклад, кількість зображень Google Earth зросла з приблизно 150 ТБ у 2006 році до 3 петабайт у 2016 році, включаючи аерофотознімки, супутникові, 2D, 3D та історичні зображення [8]. Поширення даних привело сучасне суспільство до ери великих даних із усіма послідуючими викликами [9].

Європейське космічне агентство із власними відкритими даними Sentinel [10] доповнює велику кількість безкоштовних та відкритих доступних зображень, тоді як деякі приватні компанії вкладають багато зусиль у виробництві супутникових даних [11]. Відповідним прикладом останнього випадку є PlanetScope [12], яка в даний час експлуатує 149 супутників на орбіті, із можливістю щодня відображати всю земну сушу. Навіть якщо враховувати лише оптичні супутникові знімки, Sentinel буде видавати 2 терабайти даних на день, тоді як PlanetScope буде генерувати 60 терабайт на день [13].

API Google Map – це надійний інструмент, який можна використовувати для створення власної карти, пошукової карти, функцій реєстрації, відображення даних у режимі реального часу, які синхронізуються з місцезнаходженням, планування або створення маршрутів тощо.

API Google Map є досить потужним, але ще більш кращий результат можна досягти шляхом змішування декількох API. Наприклад, поєднати в собі списки концертних програм на Картах Google і Craigslist [14]. А, отже, web-відображення даних є досить перспективним напрямком для розроблюваної інформаційної системи комплексного обслуговування авто.

Розглянувши попередні аспекти можна відмітити, що однією з головних переваг web-сайту є можливість його використання з будь-якого пристрою з підтримкою доступу в мережі Інтернет, не дивлячись на операційну систему і т.д.. Також необхідно відзначити, що технічна підтримка такої інформаційної системи є набагато простішою та менш затратною. Тому розроблювану інформаційну систему було вирішено робити web-орієнтованою з аналітичною підсистемою підбору СТО, що вигідно виділить даний ресурс на тлі існуючих конкурентів.

Прив’язка до координат за допомогою API Google Maps надасть можливість синхронізації з місцезнаходженням користувача. У свою чергу, пошук оптимального варіанту СТО можна бути здійснювати ще за рахунок розміщення клієнта.

2 ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕНЬ

2.1 Мета та задачі досліджень

У мережі Інтернет сьогодні можна зустріти безліч web-сайтів станцій технічного обслуговування. Якщо переглянути та дослідити кожен із них, то можна виявити ряд спільних рис, а саме надання інформації щодо доступних послуг, їх сильні сторони, причини віддання їм переваги серед конкурентів тощо.

Деякі додають відгуки клієнтів для відображення власної популярності серед автовласників.

При перегляданні 5-10 таких web-сайтів досить складно за короткий час обрати найзручніший та вигідний варіант СТО для звернення та отримання якісного обслуговування. Звісно існують сервіси для підбору майстерні по типам робіт.

Але при цьому необхідно відшукати та вибрати станцію, через web-сайт або телефон звернутися до них, дізнатися про час та вартість тощо.

І якщо отримана інформація не задовольняє потреби клієнта, треба повторювати даний процес заново.

У результаті автовласник може задатися питанням, чому він витрачає власний час на проведення цих операцій. СТО повинні самі бути зацікавлені в спрощені даного процесу пошуку та відбору відповідної майстерні автовласниками, підвищуючи власну конкурентоспроможність і залучаючи більшу кількість клієнтів. Саме для цього їм потрібен зручний та зрозумілий для обох сторін сервіс.

Тому, основною метою даного проекту є розробка web-орієнтованої інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО для пошуку оптимального рішення за певними критеріями користувача, що забезпечить належну організацію продуктивної взаємодії автовласників та станцій технічного обслуговування.

Для успішного досягнення поставленої мети необхідно виконати наступні задачі:

- аналіз предметної області обслуговування авто та підбору СТО;
- визначення мети та задач даного;
- аналіз вимог до проекту;
- розробка технічного завдання;
- планування робіт;
- визначення технологій реалізації;
- вибір та встановлення дизайну;
- проектування бази даних;
- розробка серверної частини;
- тестування системи;
- створення документації.

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

Даний проект також дотримується даних правил. Виходячи з цього для майбутнього створення ІС були наведені наступні функціональні вимоги:

1. Модуль реєстрації (реєстрація як клієнта, тобто власника авто, реєстрація СТО, особисті кабінети).

2. Модуль адміністрування (управління користувачами, додавання, редагування та видалення даних).

3. Модуль СТО (пошук СТО, складання заявки на ремонт: усі майстерні які підпадають під запит користувача здійснюють оцінку робіт по часу та вартості й відправляють інформацію клієнту, далі працює підсистема знаходження оптимальних варіантів і споживачу надається можливість вибору серед представлених рекомендацій).

2.2 Вибір засобів реалізації

Локальний сервер Open Server. Open Server Panel – це серверна платформа яка створена спеціально для web-розробників.

Програмний продукт має складається з набору серверних програмних засобів наряду зі зручним, функціональним та продуманим інтерфейсом [15].

Мова програмування PHP. Це серверна мова програмування, метою створення якої була генерація сторінок на стороні web-серверу. PHP є найпоширенішою мовою для її використання у сфері web-розробок. PHP існує вже досить давно та має велику підтримку. Також дана мова програмування має відкритий програмний код [16].

PHP буде знайомий усім програмістам, які працювали з такими мовами як C++ та Pascal.

Це завдяки конструкціям, які були запозичені з даних мов програмування [17].

Мова програмування JavaScript. Є досить цікавим інструментом програмування.

JavaScript застосовується для наступного:

– створення динамічних сторінок;

– JavaScript пов’язує всі блоки, будуючи так званий фундамент;

– JavaScript може динамічно здійснювати валідацію форм перед відправкою на сервер; [18].

CSS. Це технологія для опису зовнішнього вигляду HTML документа.

CSS використовується розробниками для задання шрифтів, фонових тонів, розташування елементів на сторінці тощо. Мета створення Cascading Style Sheets – розмежування зовнішнього вигляду документа з його вмістом [19].

HTML. Не є мовою програмування, а є мовою розмітки. Є формою збереження даних. HTML готує представлення даних документу для відображення у браузері. HTML розвиватися й тому є велика вірогідність використання її інструментів і надалі [20].

PHP framework Laravel. Laravel – це структура web-додатків із виразним, елегантним синтаксисом. Завдяки тому, що творець Symfony в другій версії зробив складові частини фреймворка незалежними один від одного, можна легко його розбирати на частини й збирати будь-що нове. Саме з таких деталей і був створений Laravel [21].

Це потужний інструмент для швидкої розробки, що має в собі всі необхідні складові. Laravel є простим у використанні та інноваційним фреймворком, якій підходить для вирішення широкого спектру завдань [22].

Vue.js. Створений Evan You, є прогресивною структурою JavaScript з відкритим кодом для побудови користувальницьких інтерфейсів (UI) та односторінкових додатків. Фреймворк дозволяє розробникам поступово створювати користувальницькі інтерфейси [23].

Крім того, Vue доступний ряд модулів, які реалізують сучасний підхід до розробки web-додатків [24]. Сильною стороною Vue.js. є наявність зрозумілої документації [25].

IDE PHPStorm. Програмне забезпечення JetBrains PhpStorm включає в себе засоби для розробки web-додатків [26].

У процесі використання IDE є можливість знаходження корисного поєднання клавіш і інші закладені розробниками функції [27].

PHPStorm також має деякі з них, а саме:

- автогенерація коду;
- створення PHPDoc;
- автоформатування коду;
- підказки (Code Completion);
- консоль;
- різновиди пошуку.

Git. Це система контролю версій коду. Створена для полегшення процесу розробки програмних продуктів у командах. На даний момент є найпоширенішою у світі. Завдяки Git програмісти обмінюються своїми проектами тим самим допомагаючи один одному. [28].

Переваги:

– Git має всі плюси використання VCS, що і в Subversion;

– має легкий процес шифрування «із коробки»;

– у разі припинення роботи сервера з головним репозиторієм, можна робити коммітів в локальний сервер і чекати відновлення роботи сервера [29];

Недоліки:

– команди Git, орієнтовані на набори змін, а не на файли, можуть викликати здивування у користувачів, які звикли до файл-орієнтованим VCS, таким як SVN;

– використання для ідентифікації ревізій хеш SHA1, які призводить до необхідності оперувати довгими рядками замість коротких номерів версій, як у багатьох інших системах;

– великі витрати часу, у порівнянні з файл-орієнтованими системами, на формування історії конкретного файлу;

– система не вміє відслідковувати порожні каталоги;

– деякі команди працюють несподівано, зокрема, можуть призводити до неочевидним помилок [30].

Отже, інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО вирішено реалізовувати з використанням сучасних web-технологій.

За серверну частину відповідатиме мова програмування PHP та фреймворк Laravel, клієнтську частину – JavaScript та фреймворк Vue.js. База даних буде представлена через MySql.

Сучасна web-розробка не може бути виконана без використання систем контролю версій. У даній роботі буде застосовано Git, а для зручності написання коду – IDE PHPStorm.

Розробку вирішено вести на локальному сервері OpenServer.

3 ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Структурно-функціональне моделювання

Структурно-функціональне моделювання – це процес побудови окремих процесів об’єкта автоматизації. Воно реалізується у вигляді так званих ієрархічних пов’язаних між собою діаграм. Описуються компоненти системи з певною декомпозицією.

Першим кроком створюються загальна діаграма. Там міститься загальний опис системи.

Наступним кроком загальна діаграма певним чином деталізується.

IDEF0 – нотація для графічного представлення для створення функціональних моделей та відображення структури та функцій системи.

Реалізація функціональної моделі була в обраному програмному продукті Erwin Process Modeler.

Контекстною діаграмою називають діаграму, в якій об’єкт моделювання представлений однорідним блоком із граничними стрілками. Вони відображають певний зв’язок об’єкта із середовищем існування.

Контекстна діаграма процесу роботи даної інформаційної системи представлена у рисунку 3.1.

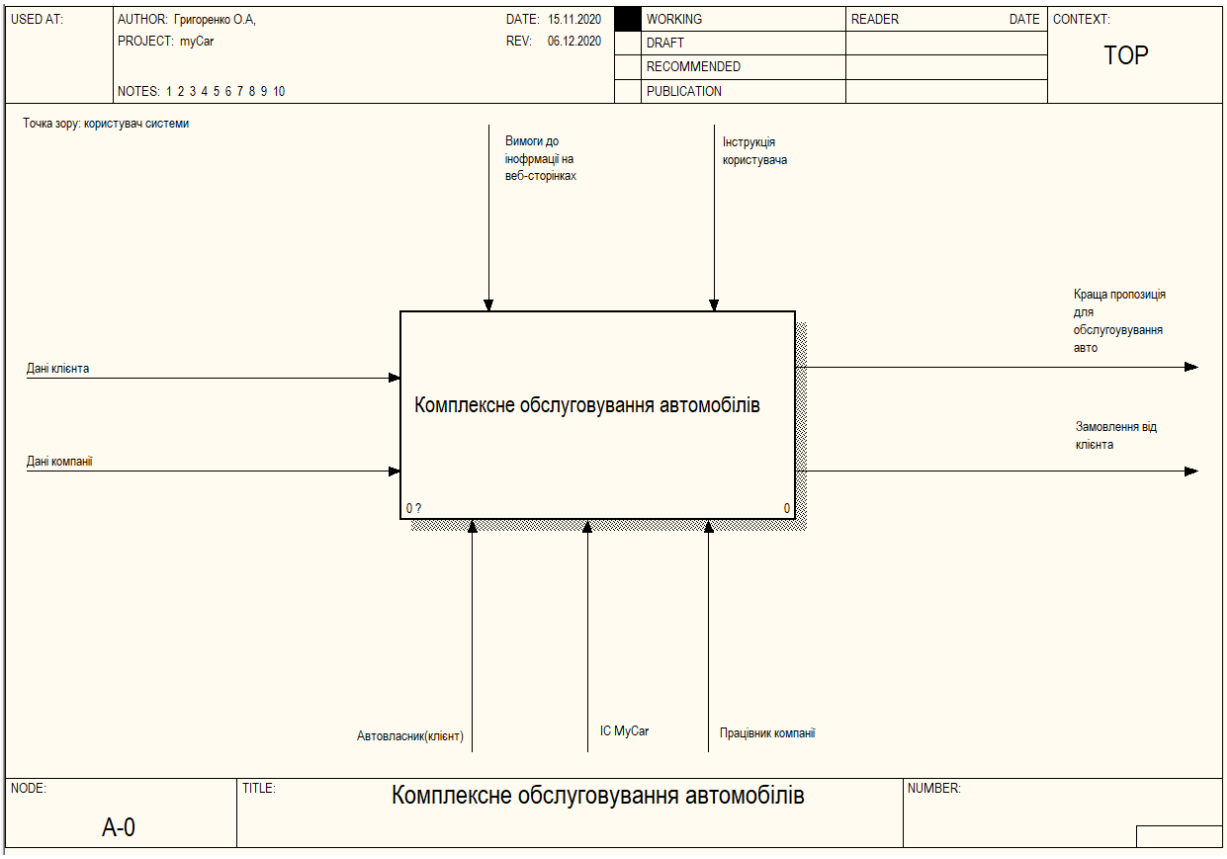


Рисунок 3.1 – Контекстна діаграма процесу

Також дана нотація має можливості поступового розпису головного процесу. Для цього створюються дочірні діаграми, які оплюють однакову область с батьківським процесом.

Дочірні діаграми описують процес детальніше. Спираючись на дану нотацію стрілки батьківського процесу переносяться на діаграму нижчого рівня граничними стрілками.

Виділяються такі типи стрілок, як «Вхід», «Вихід», «Механізм», «Управління».

Механізми ідентифікують засоби, які підтримують виконання процесу. Таким чином, блок IDEF0 показує перетворення входу у вихід за допомогою механізмів із урахуванням керуючих впливів.

Виходи – дані або матеріальні об’єкти, вироблені процесом. Вхідні дані перетворюються або витрачаються процесом, щоб створити те, що отримаємо на його виході.

Декомпозиція головної діаграми наведена у рисунку 3.2.

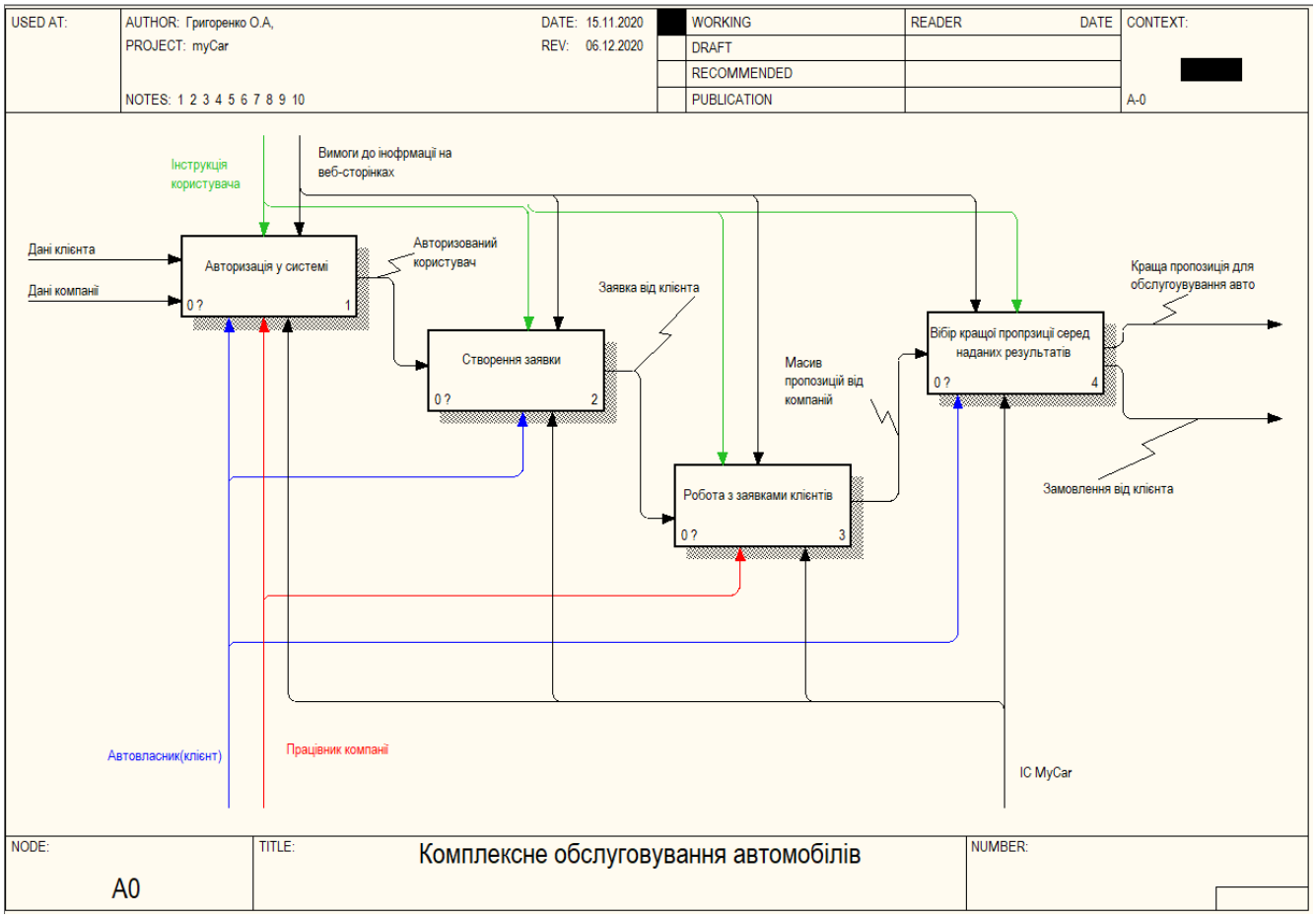


Рисунок 3.2 – Декомпозиція контекстної діаграми

Наступним кроком більш деталізовано описуються процеси, які потребують конкретики та детального розгляду.

Декомпозиція процесу авторизації та реєстрації продемонстровано на рисунку 3.3. Декомпозиція процесу роботи з заявками та вибору оптимальної пропозиції (СТО) наведено на рисунках 3.4-3.5.

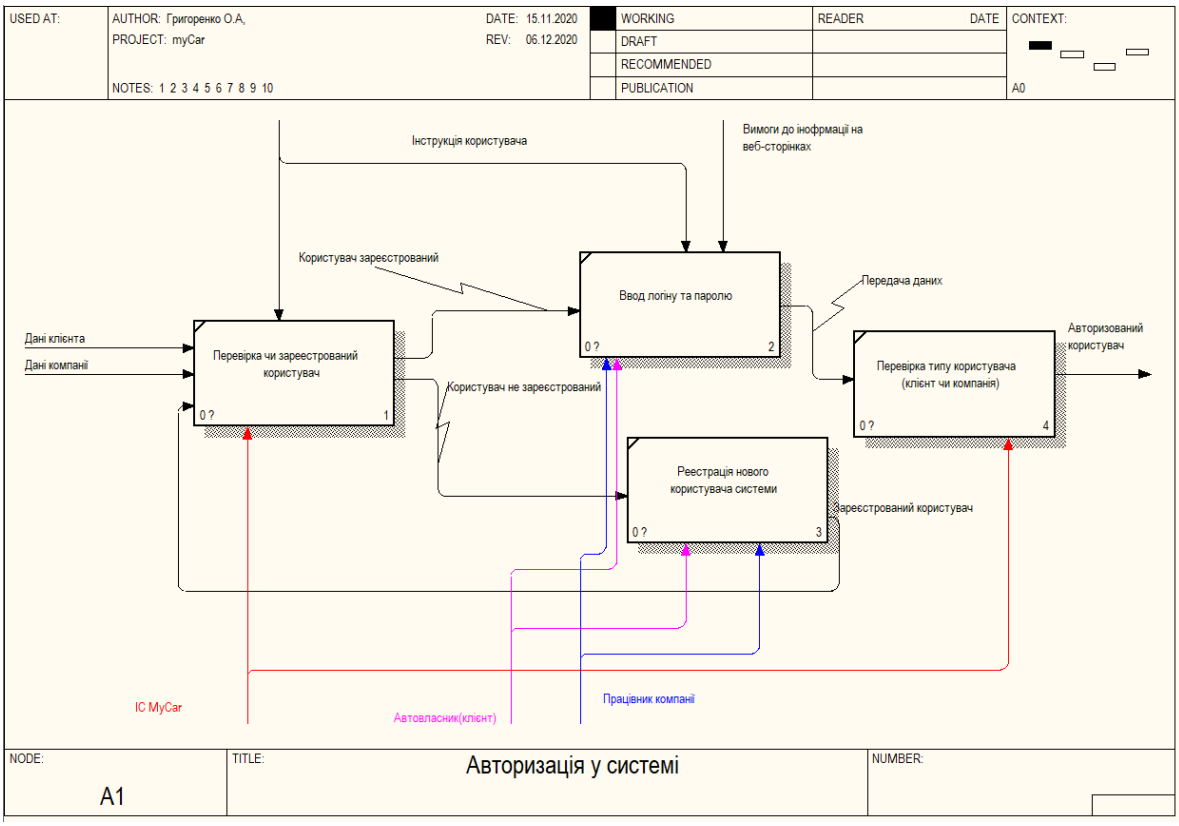


Рисунок 3.3 – Декомпозиція процесу авторизації

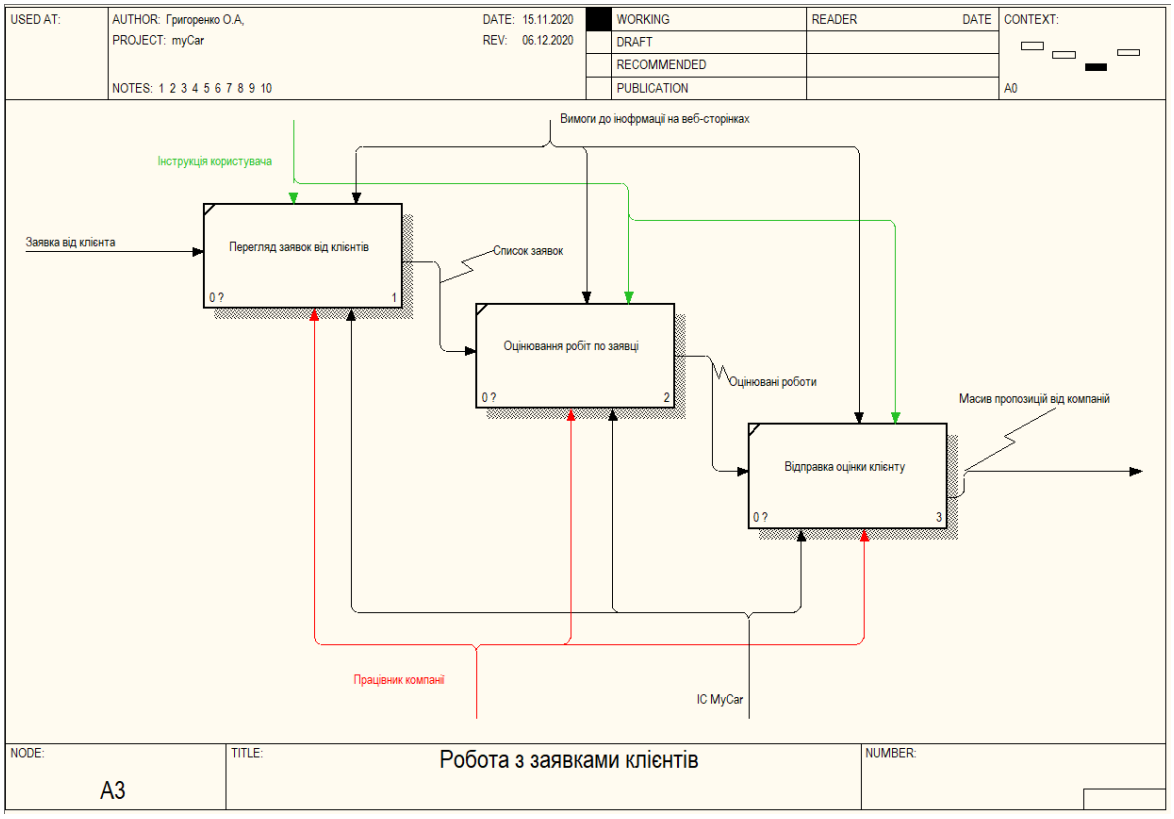


Рисунок 3.4 – Декомпозиція процесу роботи з заявками

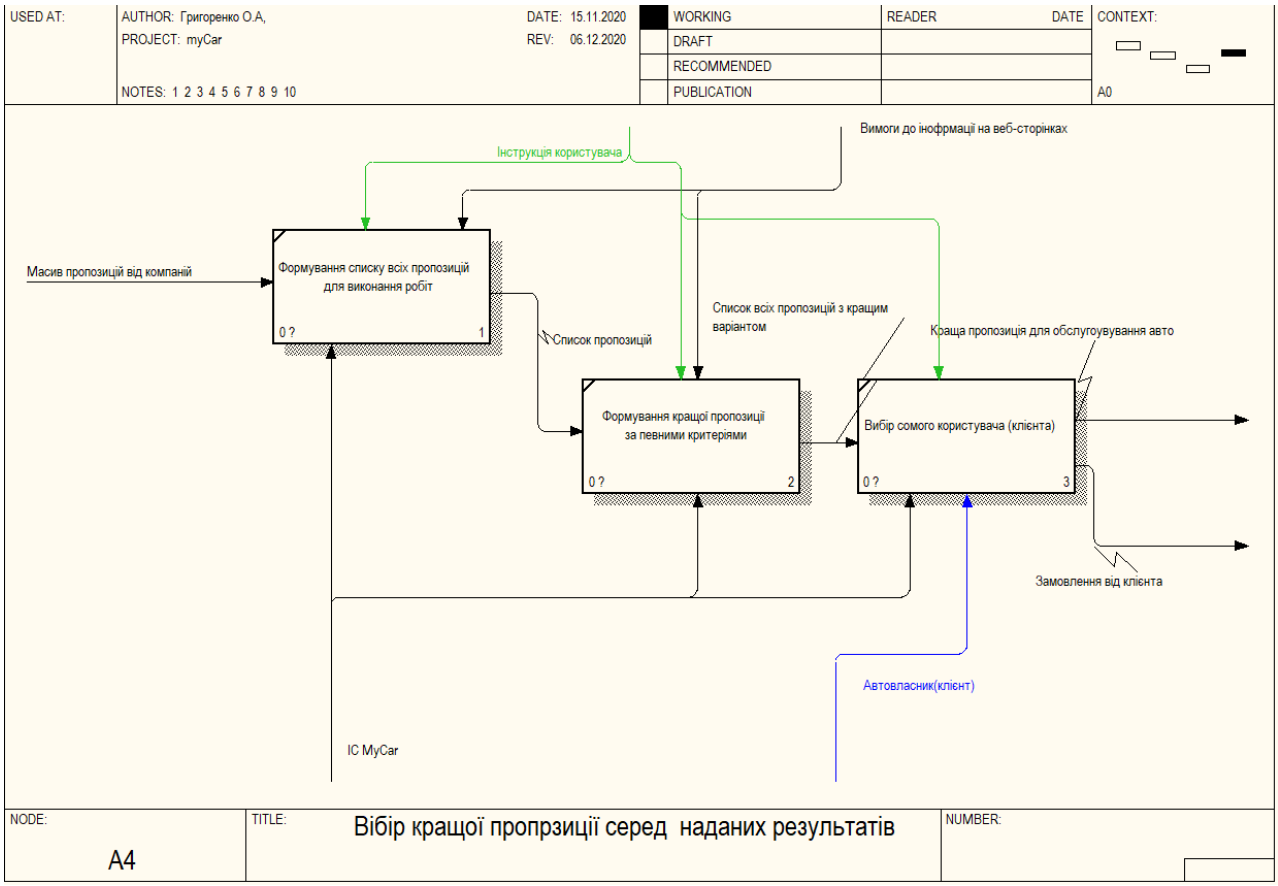


Рисунок 3.5 – Декомпозиція процесу вибору кращої пропозиції

3.2 Створення та проектування бази даних

База даних складається з певної кількості даних із конкретної предметної галузі, які зберігаються в пам’яті.

Система управління базами даних (СУБД) повинна надавати певні механізми пошуку даних за спеціальними стандартизованими запитами, має забезпечувати захист даних від несанкціонованого доступу.

Інформація в базах даних з реляційною моделлю зберігається у вигляді таблиць. Може створюватися безліч таблиць у базі даних. Це обмежується тільки вільним місцем у пам’яті.

В процесі роботи з декількома таблицями мають встановлюватися певні зв’язки.

При створенні БД необхідно детально розробити її структуру.

Цей процес включає наступні етапи:

– визначення мети для чого створюється база ;
– визначення певних таблиць, які база даних повинна містити;
– визначення для кожної таблиці списку полів;
– визначення ключових полів з унікальними значеннями;
– визначення зв’язків між таблицями;
– введення даних;

База даних повинна задовольняти вимогам тих користувачів, які будуть з нею працювати.

Спочатку необхідно визначити, які функції буде виконувати база даних та яка інформація буде міститись.

Схема даних розроблюваної інформаційної системи представлена на рисунку 3.6.

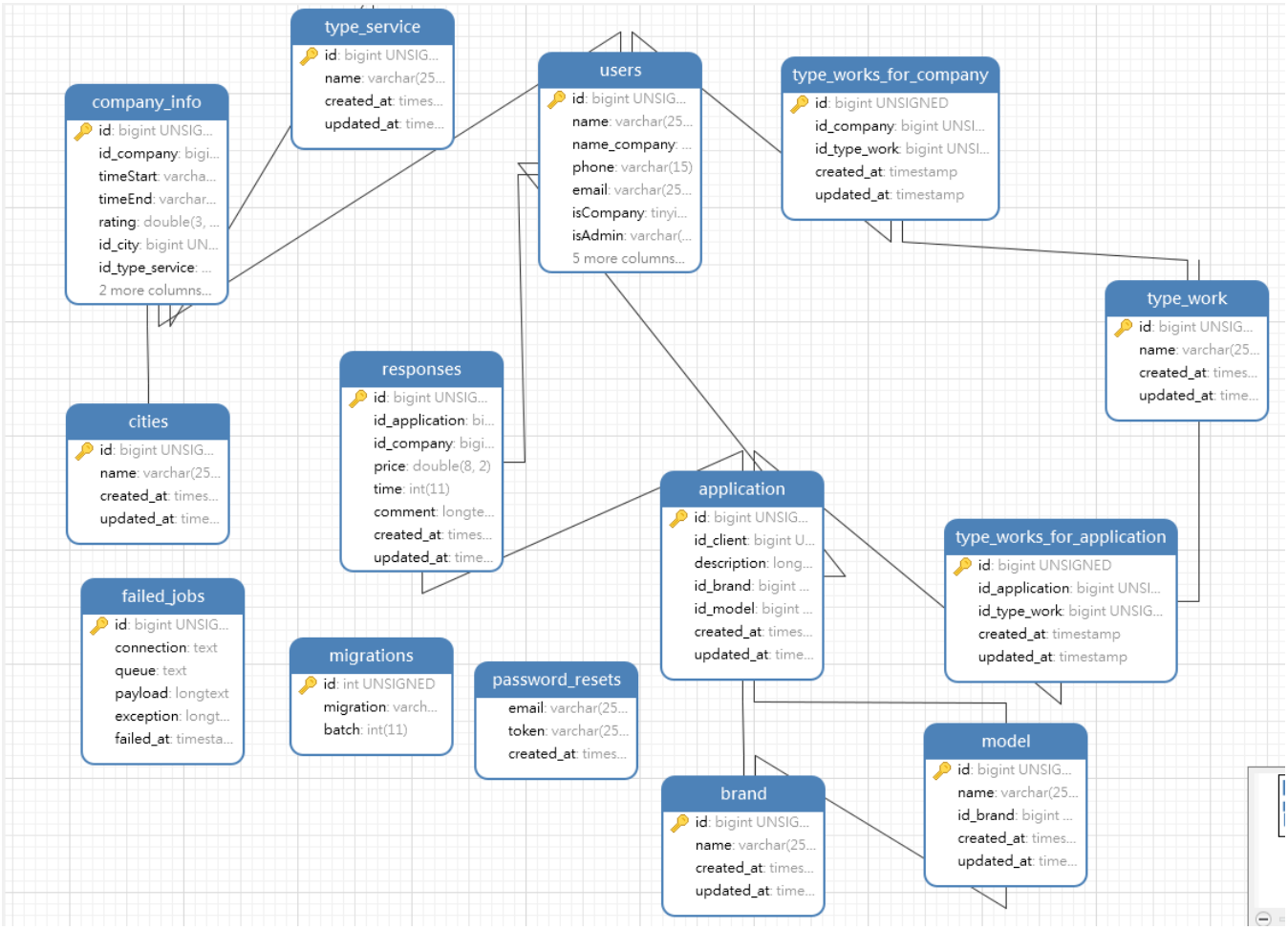


Рисунок 3.6 – Структура бази даних

Більш детально про сутності та їх значення таблиць бази даних описано 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

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

Продовження таблиці 3.1 - Опис сутностей

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
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

3.3 Моделювання Use Case діаграми

Use Case діаграма або модель варіантів використання дає більш точне уявлення про те як працює та чи інша система.

Дана діаграма робить опис варіантів взаємодії між учасниками. Цих учасників може бути 1 і більше.

Учасником може бути людина або інша інформаційна система.

Моделі варіантів використання складається з опису акторів, варіантів використання (ВВ) програмного продукту (ПП) та діаграми варіантів використання. Use Case діаграма розроблюваної інформаційної системи (ІС) представлена на рисунку 3.7.

Актори Use Case

Адміністратор – переглядає користувачів, має можливість редагувати їхні дані та видаляти за необхідності.

Автовласник – клієнт, який створює заявку на обслуговування авто, отримує всі варіанти можливих СТО за зазначеними критеріями, включаючи оптимальний та обирає кращий.

Працівник СТО – співробітник компанії, який оцінює заявки на ремонт та відправляє клієнту.

Варіанти використання

ВВ Авторизація у системі – дає можливість користувачу працювати з певним рівнем доступу та інтерфейсу.

ВВ Реєстрація – надає можливість реєструватися.

ВВ Редагування/видалення користувачів – надає можливість адміністратору системи змінювати або видаляти дані користувачів.

ВВ Створення заявки – дозволяє клієнту створювати заявки на ремонт.

ВВ Оцінка заявки – дозволяє робітнику приймати заявки, які відповідають специфіці організації та оцінювати роботи й відправляти клієнту.

ВВ Вибір кращого варіанту – алгоритм системи, який обирає оптимальний варіант на основі певних критеріїв.

Рисунок 3.7 – Діаграма варіантів використання

3.4 Вибір алгоритму для пошуку оптимального рішення

Ураховуючи поставлені цілі, а саме вибір кращої пропозиції від станцій технічного обслуговування, можна зробити висновок, що на вході маємо певний масив із пропозиціями від СТО. Завдання полягає в тому, щоб з даного масиву обрати варіант, який найбільше підходить за наступними критеріями:

1. Вартість послуг (кожна людина прагне отримати кращий результат за меншу ціну).
2. Час на виконання робіт.
3. Рейтинг СТО.

Аналізуючи існуючі алгоритми було обрано за основу «Жадібний алгоритм» – це алгоритм, який на кожному кроці робить локально найкращий вибір в надії, що підсумкове рішення буде оптимальним.

На вході існує деякий масив усіх пропозицій по певній заявці.

Алгоритм пошуку оптимальної пропозиції працює за наступним чином:

1. Встановлюється вага критеріїв на основі аналітичних даних.

2. Виконується сортування масиву пропозицій по кожному з критерії та обріється кращий.

3. Виконується порівняння результатів.

Схема пошуку оптимальної пропозиції наведена на рисунку 3.8.

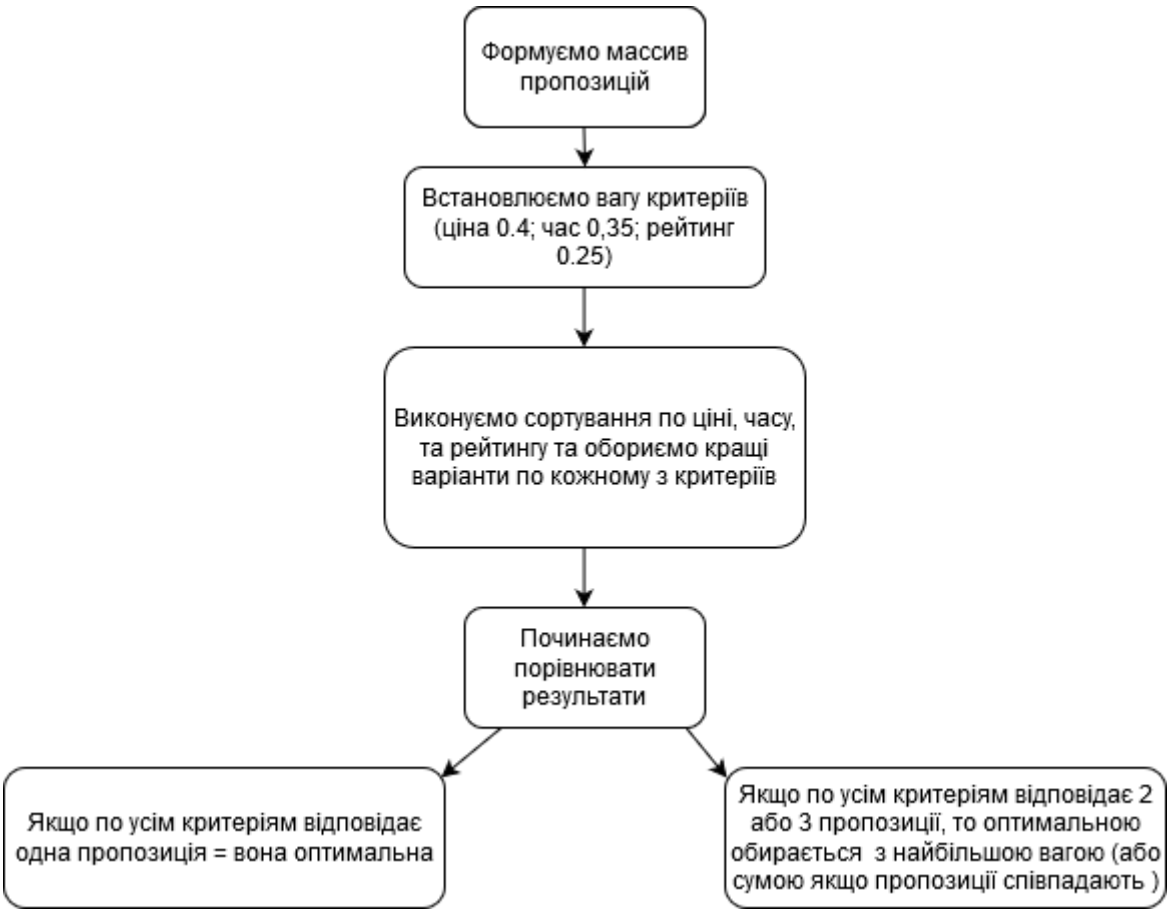


Рисунок 3.8 – Схема пошуку оптимальної пропозиції

4 ПРОЦЕС РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Встановлення необхідних інструментів та їх налаштування

Система комплексного обслуговування автомобілів згідно поставленої задачі має розроблятися у вигляді веб-додатку.

Виходячи з цього для початку реалізації необхідно встановити локальний веб-сервер і базу даних.

На платформі Windows оптимальним варіантом для розробки є OpenServer. Наявність всіх необхідних модулів та широкий вибір підтримуваних версій мови PHP надають розробнику комфортні умови для локальної розробки програмного продукту.

Налаштування серверу OpenServer показано на рисунку 4.1.

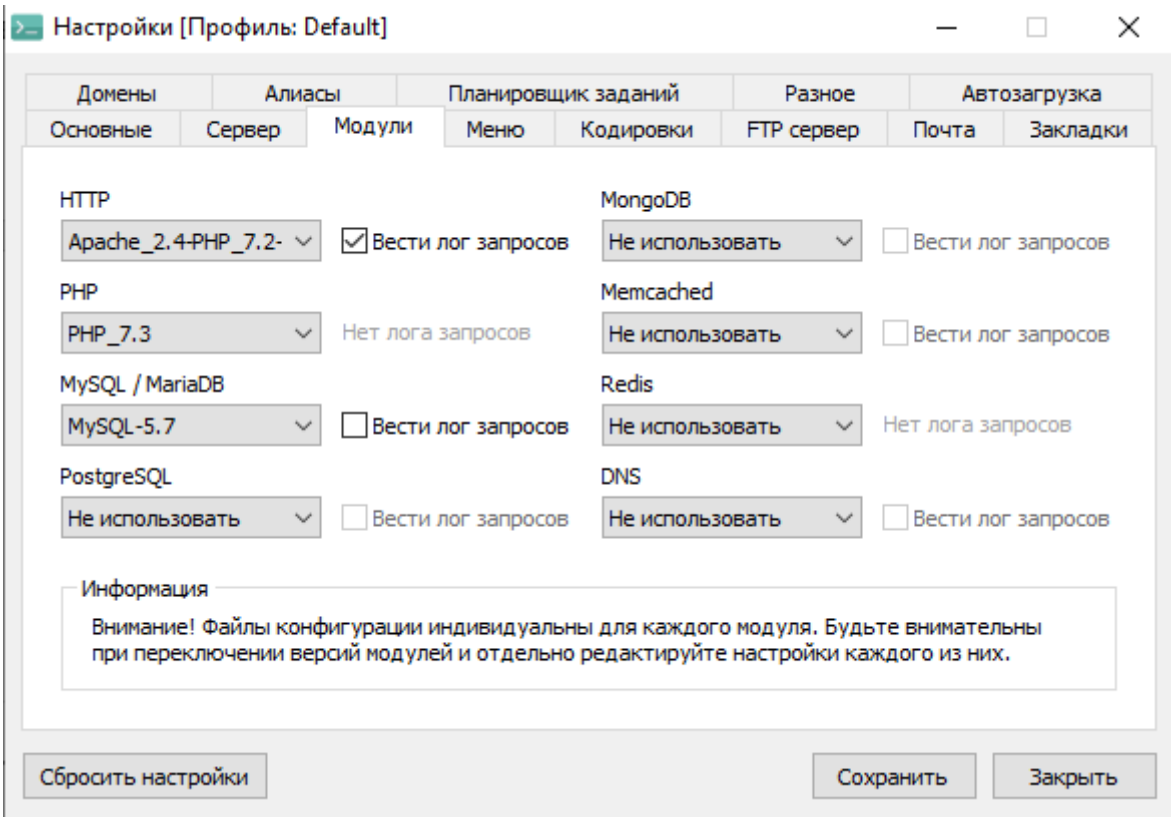


Рисунок 4.1 – Налаштування серверу OpenServer

Дана інформаційна система була створена на основі Laravel. Це гарне рішення для швидкого та вмілого створення безпечного та надійного веб-додатку. Він працює з використанням MVC. Дана схема наведена на рисунку 4.2. З неї видно, що вона складається з декількох ключових елементів, а саме:

1. Модель – зв’язується з базою даних та надає необхідні дані.
2. Вид – відображає отримані дані від моделі та оброблені контролером.
3. Контролер – виконує обробку даних отриманих з моделі.

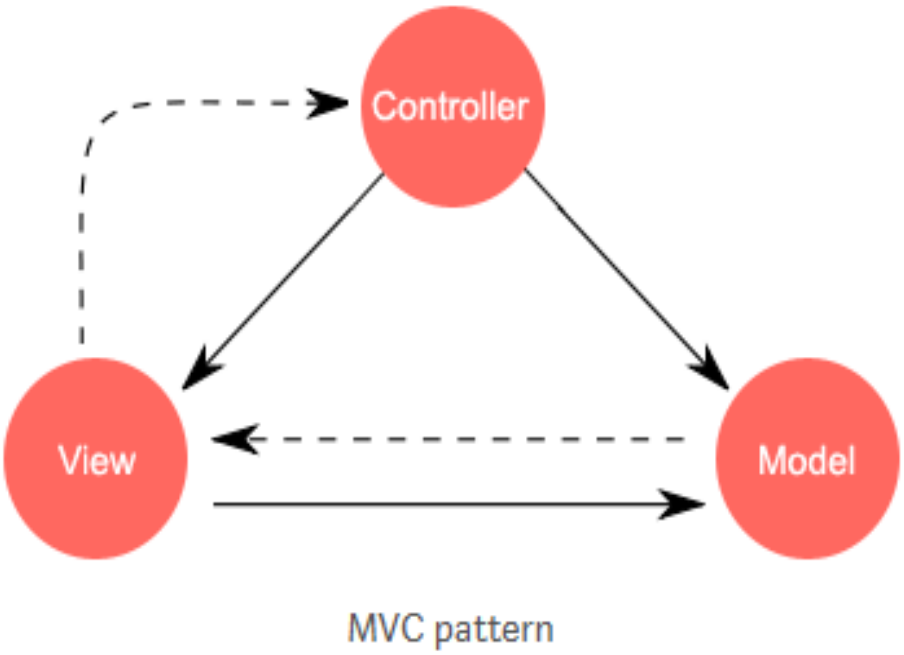


Рисунок 4.2 – MVC

Перед роботою з Laravel необхідно встановити Composer. Він потрібен для завантаження необхідних моделей, у тому числі й самого Laravel.

Щоб перевірити роботу Composer необхідно в командному рядку необхідно ввести команду composer.

Отриманий результат повинен бути такий, як на рисунку 4.3.

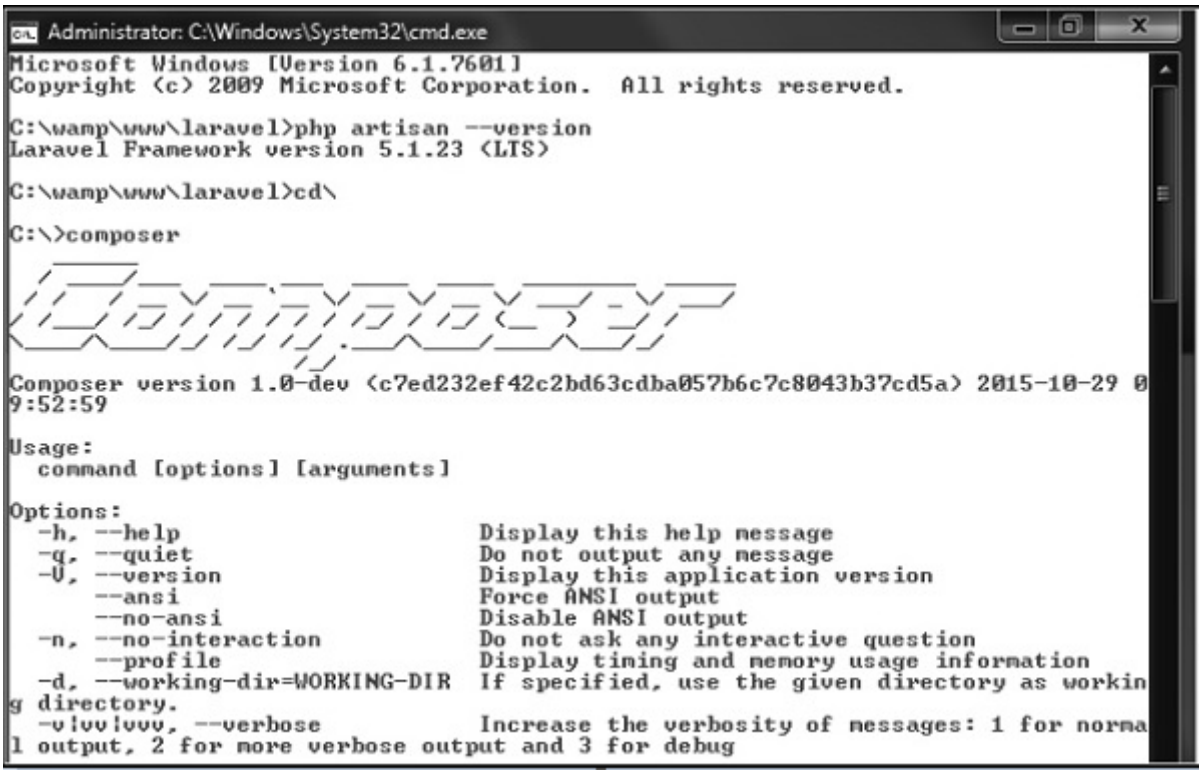


Рисунок 4.3 – Перевірка працездатності Composer

Якщо в ході перевірки все добре далі встановлюється сам фреймворк. Це робиться за допомогою команди:

```
composer global require "laravel/installer"
```

Composer завантажить всі необхідні файли Laravel у вибрану директорію.

Після розпакування файлів в обраному каталозі стали доступними необхідні директорії як показано на рисунку 4.4.

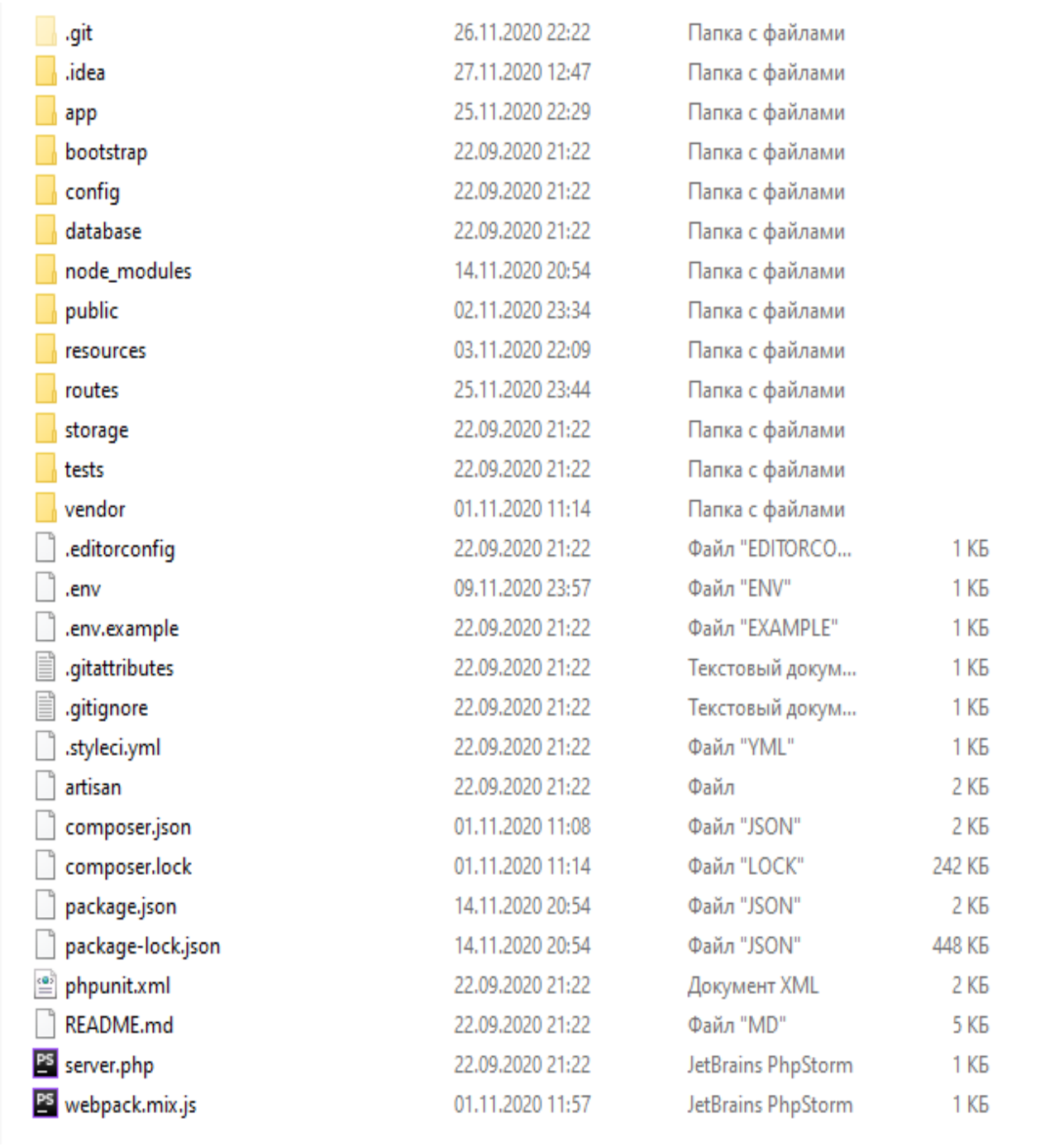


Рисунок 4.4 – Коренева директорія Laravel

Laravel має наступну структуру:

– app – даний каталог містить логічну структуру додатку;

– bootstrap – дана директорія необхідна для ініціалізації додатку.

– public – має в собі точку входу та публічні файли, такі як зображення;

– vendor – дана директорія одна з найважливіших.

– Там знаходяться файли самого фреймворку так і додаткові бібліотеки, які використовуються.

Для комфортної розробки веб-додатків потрібен зручний та функціональний текстовий редактор. Таким редактором є PhpStorm.

Безкоштовну версію на 30 діб можна завантажити з офіційного сайту (https://www.jetbrains.com). Встановлюється як звичайна Windows програма без зайвих проблем.

Після встановлення на персональний комп'ютер PhpStorm пропонує провести початкове налаштування.

Головну робочу область текстового редактору наведено на рисунку 4.5

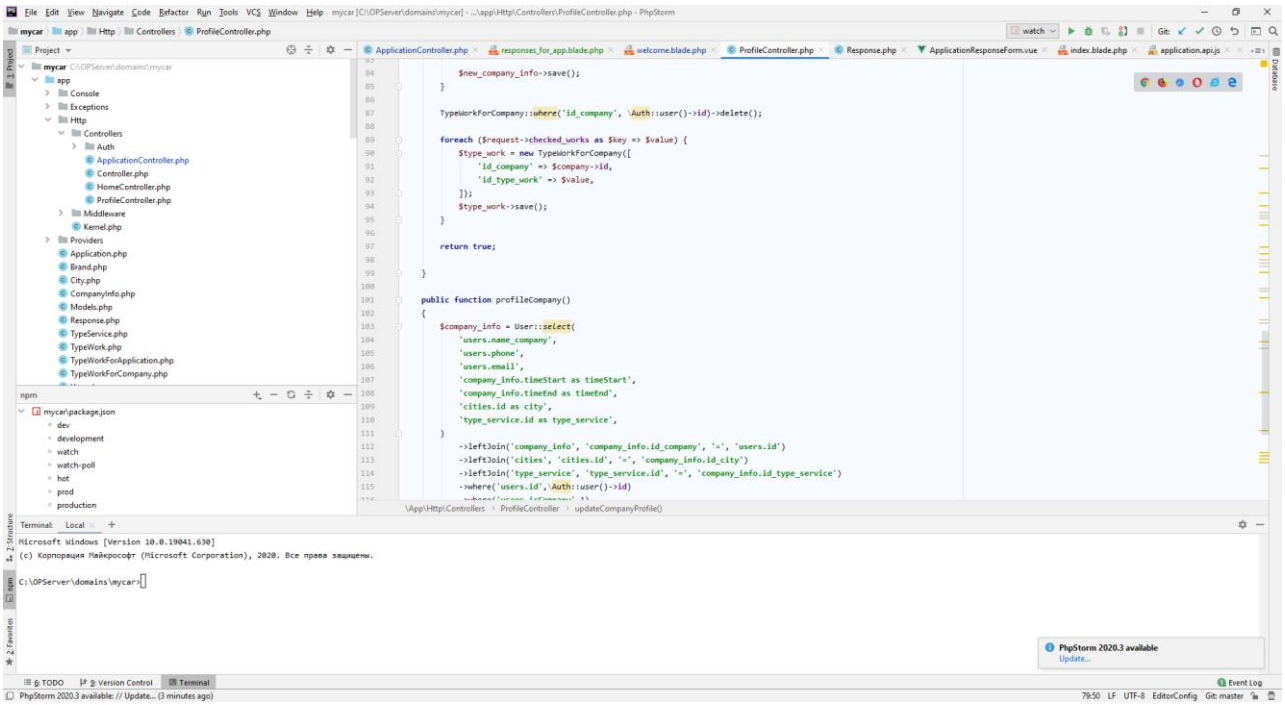


Рисунок 4.5 – Головне вікно PhpStorm

Є велика вірогідність виникнення ситуацій, в яких потрібно відмінити зміни в коді та встановити більш ранню версію додатку. Для уникнення збереження всього проекту окремо кожен раз при додаванні змін було створено Git. Це система для контролю версій коду.

Git встановлюється як звичайний Windows додаток. Встановлення продемонстровано на рисунку 4.6. Git – це консольна програма та користування нею в консольному режимі є незручним. Тому для полегшення роботи з Git потрібно додатково встановити графічну програму SmartGit (рис. 4.7).

Рисунок 4.6 – Git

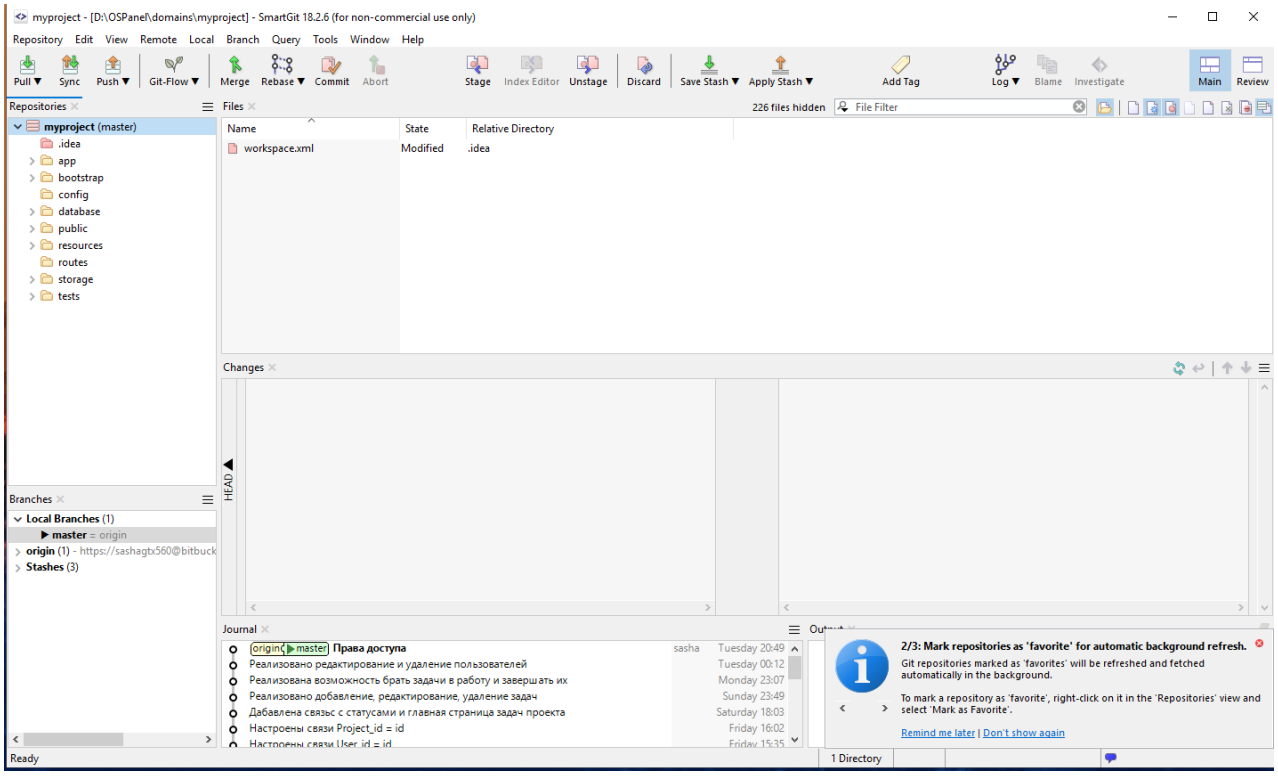


Рисунок 4.7 – SmartGit

Git зберігає дані у віддалений репозиторій. Найбільш популярними в мережі такими репозиторіями є GitHub та Bitbucket. Оскільки їхній функціонал є майже однаковий, було обрано Bitbucket (рис. 4.8).

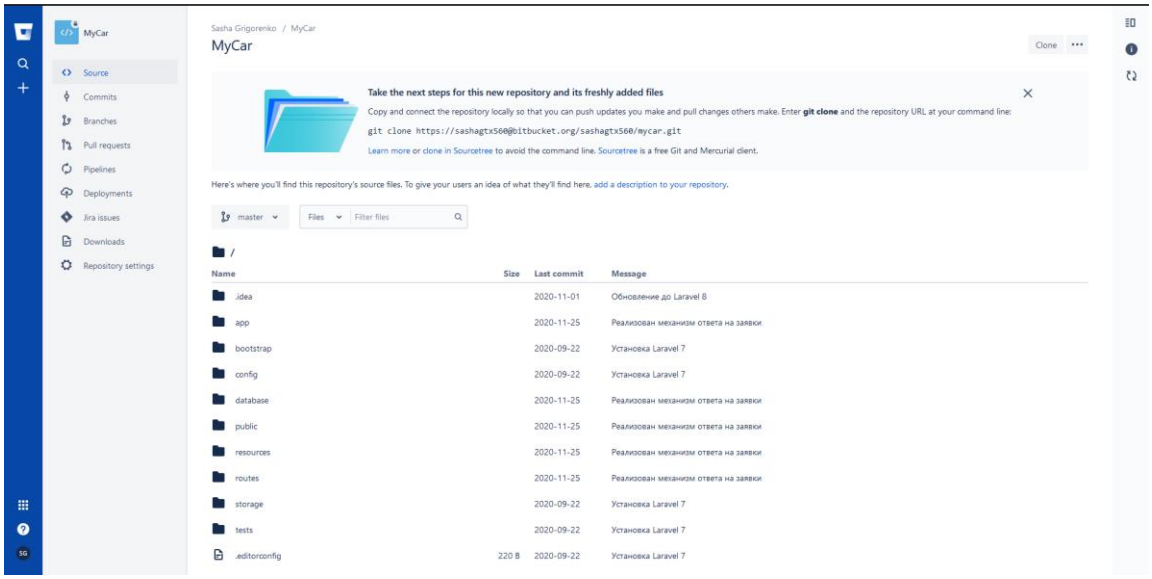


Рисунок 4.8 – Bitbucket

Для коректної роботи Bitbucket та SmartGit потрібно їх зв’язати. Для цього у Bitbucket спочатку було створено новий проект та скопійовано SSH ключ. Даний ключ треба ввести в налаштування SmartGit, як це показано на рисунку 4.9.

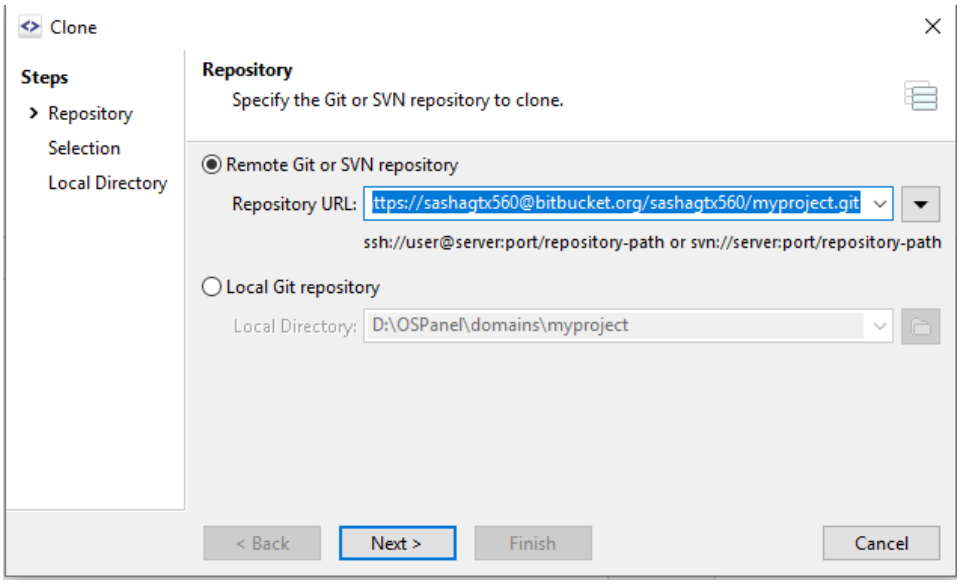


Рисунок 4.9 – Налаштування доступу SmartGit

4.2 Програмна реалізація інформаційної системи

Кожен з ключових етапів розробки потрібно коментувати та відправляти до віддаленого репозиторію Bitbucket через програму SmartGit. Це потрібно для того, щоб в будь-який момент можна було переглянути зроблені зміни в коді та якщо є необхідність повернутися до попередньої версії. Вікно зроблених коментарів у SmartGit наведено у рисунку 4.10.

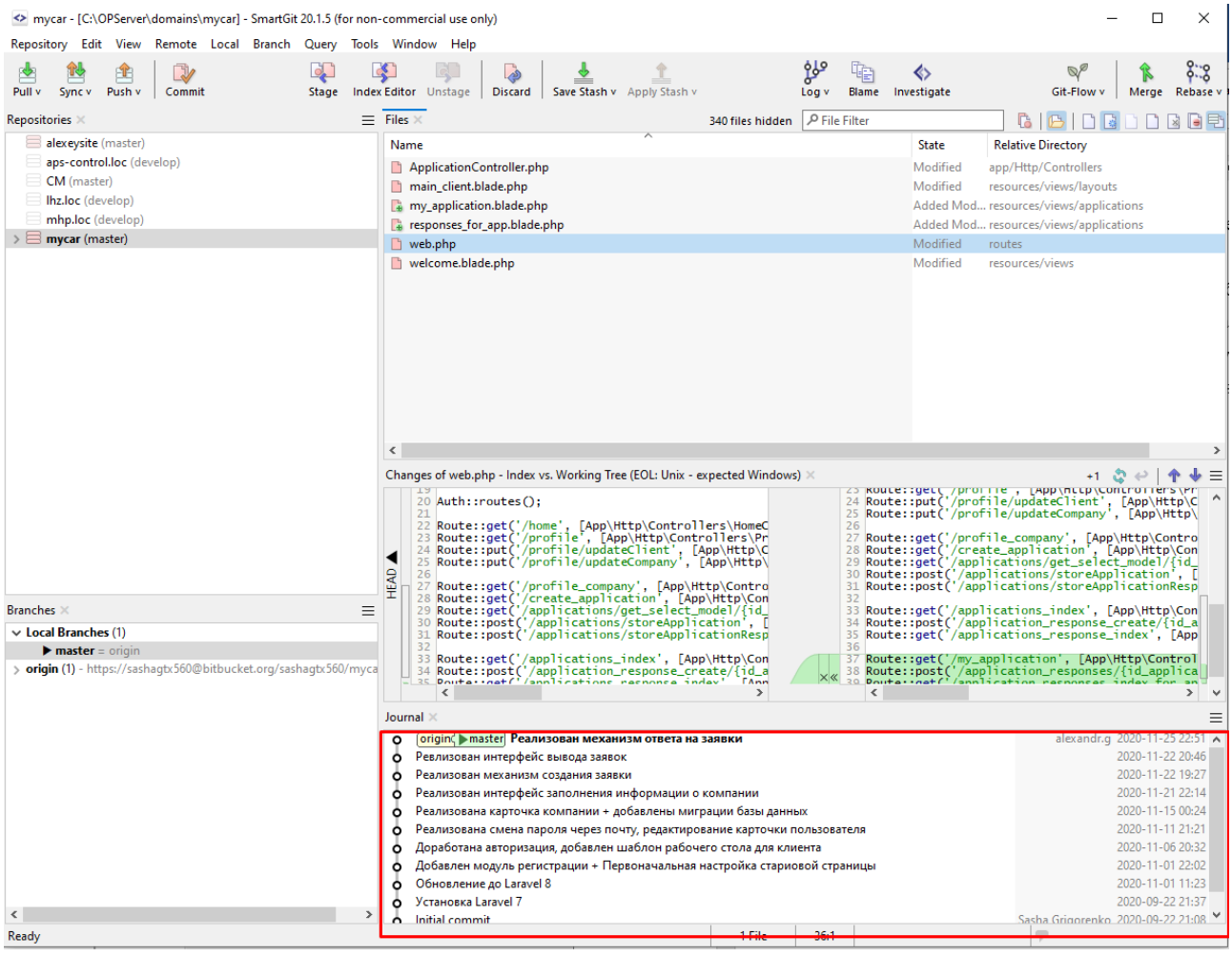


Рисунок 4.10 – Коментарі розробки

Реалізація інформаційної системи на основі Laravel почалася з налаштування параметрів у файлі **.env**.

У даному файлі зберігаються налаштування для підключення бази даних, режиму розробки та поштових налаштувань.

Файл налаштувань наведено на рисунку 4.11.

```
APP_NAME=MyCar
APP_ENV=local
APP_KEY=base64:ybdF4yUcwCejw+wirxlt/WObPIe9taXog9hQBw5yIT8=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3307
DB_DATABASE=mycar
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=technotestlep@gmail.com
MAIL_PASSWORD=
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=technotestlep@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```

Рисунок 4.11– Файл налаштувань .env

Наступним кроком необхідно було створити базу даних на локальному сервері та за допомогою вбудованих можливостей Laravel створити таблиці з необхідними полями та зв’язках між ними, що було реалізовано за допомогою «міграцій». Це вбудований механізм Laravel для спрощення взаємодії з базою даних.

За допомогою файлів міграції, які наведені на рисунку 4.12, прописуються поля таблиць, їхній тип даних та необхідні зв’язки.

Далі за допомогою команди `php artisan migrate` відбувається збереження налаштувань в базу даних.

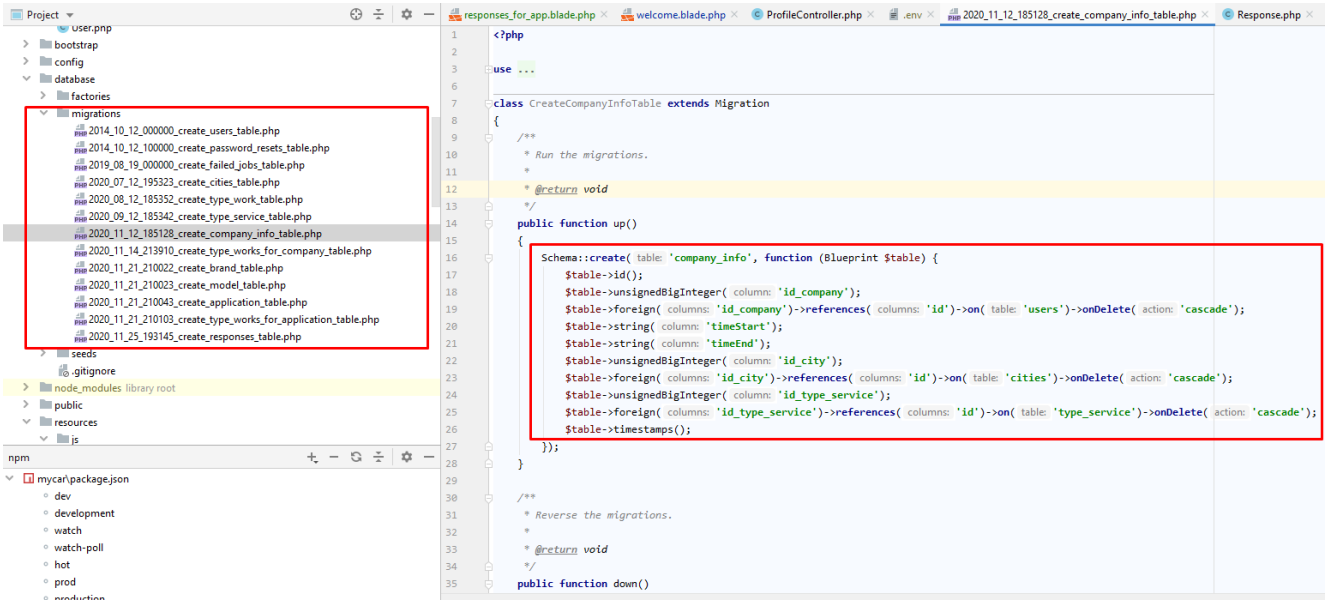


Рисунок 4.12 – Структура файлів міграцій

Після створення структури БД необхідно було додати механізми для звернення до бази даних з PHP. Так як використовується фреймворк Laravel, а він працює на паттерні MVC, таким механізмом є модель. Остання забезпечує підключення до таблиці бази даних. Також там вказується поля доступні до редагування. Структура файлу моделі наведена на рисунку 4.13.

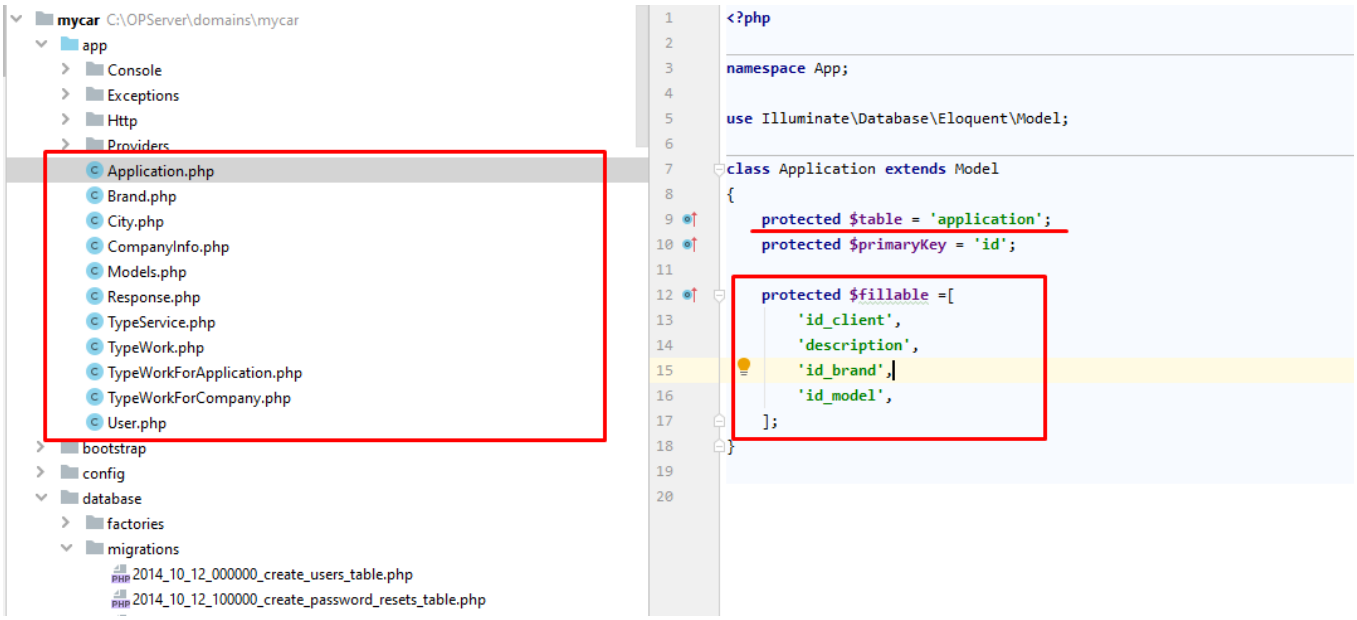


Рисунок 4.13 – Структура моделей

Для створення логіки роботи з даними треба було задати контролери, в яких відбувається збір необхідних даних та передача їх до шаблонів відображення.

Приклад контролера наведено на рисунку 4.14. Детальний програмний код наведено у Додатку Б.

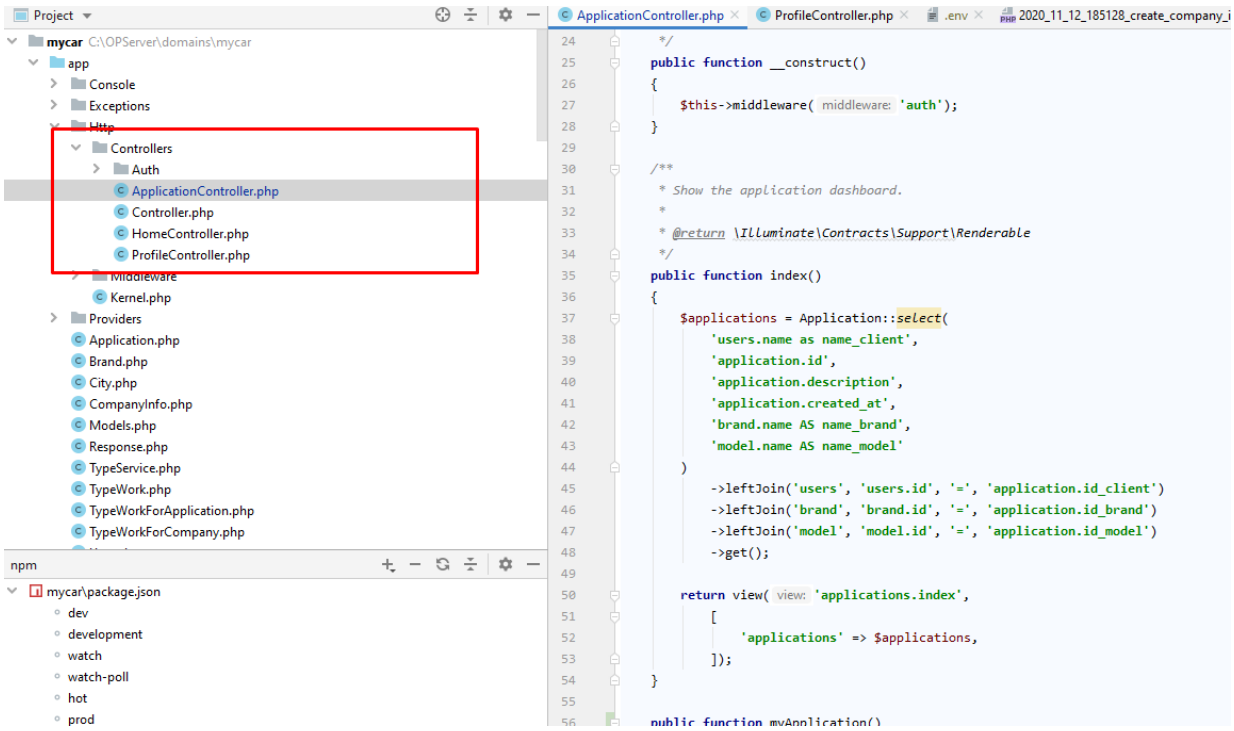


Рисунок 4.14 – Структура контролера

Після того, як дані отримані, їх необхідно відправити до HTML шаблонів. У Laravel використовується власний шаблонізатор blade. Він дещо спрощує взаємодію HTML та PHP. Приклад шаблона наведено на рисунку 4.15. Детальний програмний код наведено у Додатку Б.

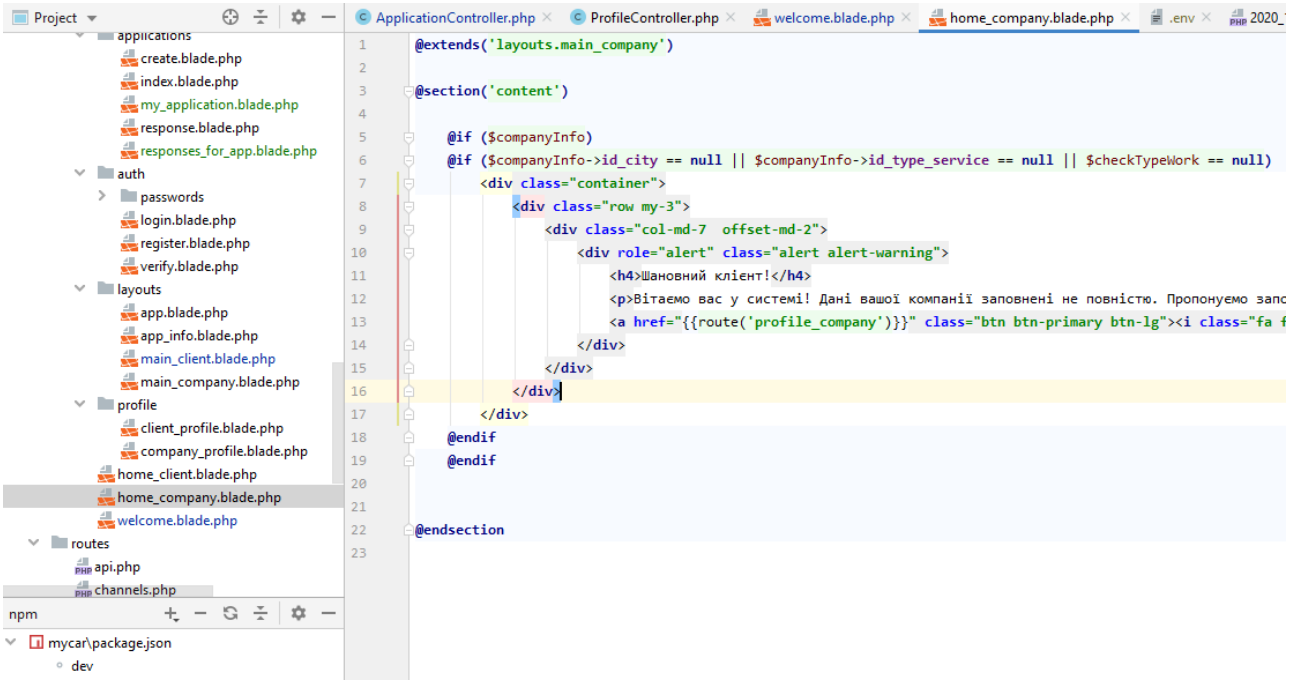


Рисунок 4.15 – Шаблон blade

Окрім HTML та CSS для динаміки сторінок використовувався фреймворк Vue.js. Він працює на основі компонентів, які взаємодіють між собою. Компоненти вбудовуються в шаблони і через них до компоненту передаються необхідні дані з контролеру. Приклад інтеграції компонента наведено на рисунку 4.16.

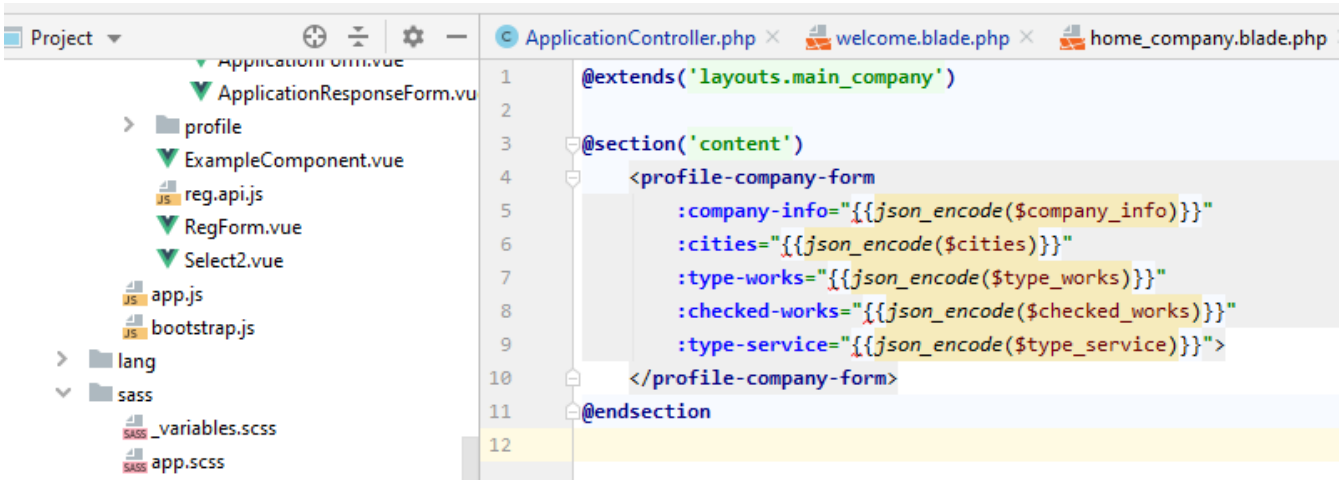


Рисунок 4.16 – Інтеграція компоненту Vue.js

Компоненти Vue.js містять в собі HTML розмітку, JS блок для створення логіки та CSS блок для змінення стилів. Детальний програмний код наведено у Додатку Б.

4.3 Демонстрація роботи розробленої інформаційної системи

Веб-орієнтована інформаційна система «MyCar» зустрічає користувача головною сторінкою, яка наведена у рисунку 4.17.

На сторінці наведені основні пункти меню навігації. Головними з них є вхід до системи та реєстрація.

При натисканні на реєстрацію буде відкрита відповідна форма (рис.4.18).

Реалізована можливість реєстрації як власника авто, так й організації. Для цього реалізований спеціальний перемикач, який динамічно змінює форму (рис. 4.19).

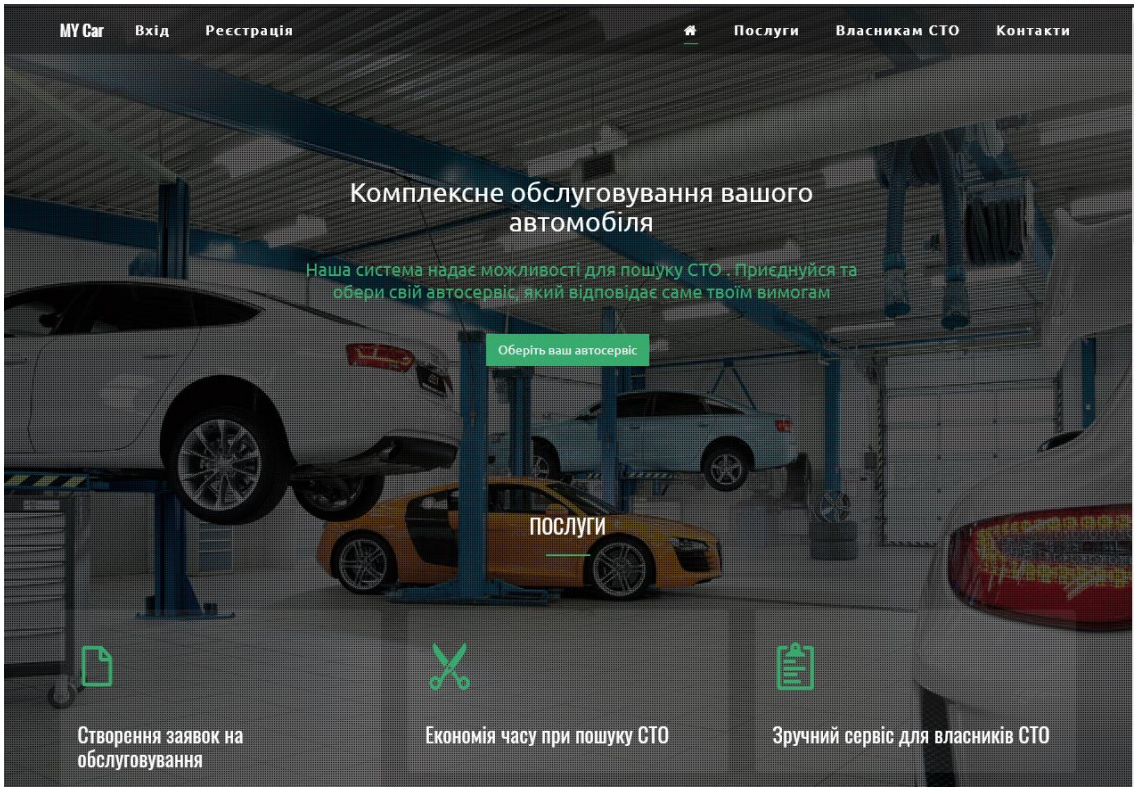


Рисунок 4.17 – Головна сторінка

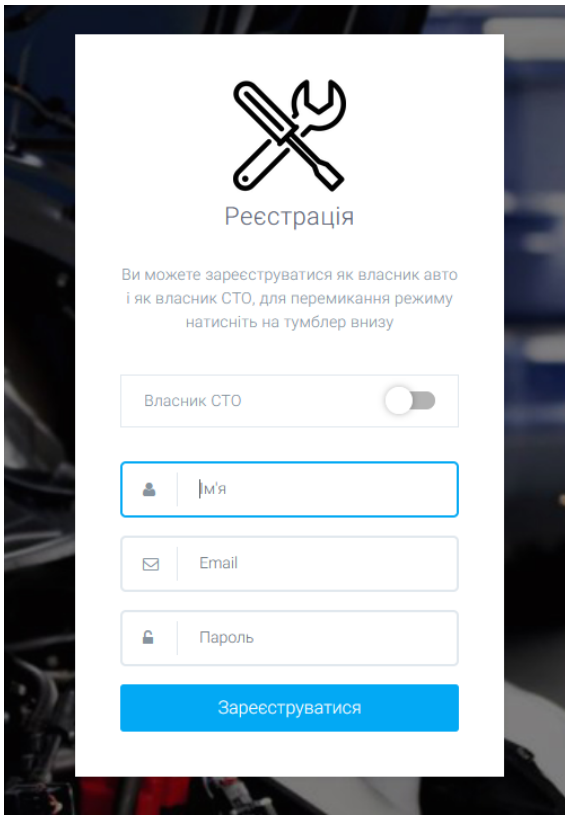


Рисунок 4.18 – Форма реєстрації клієнта

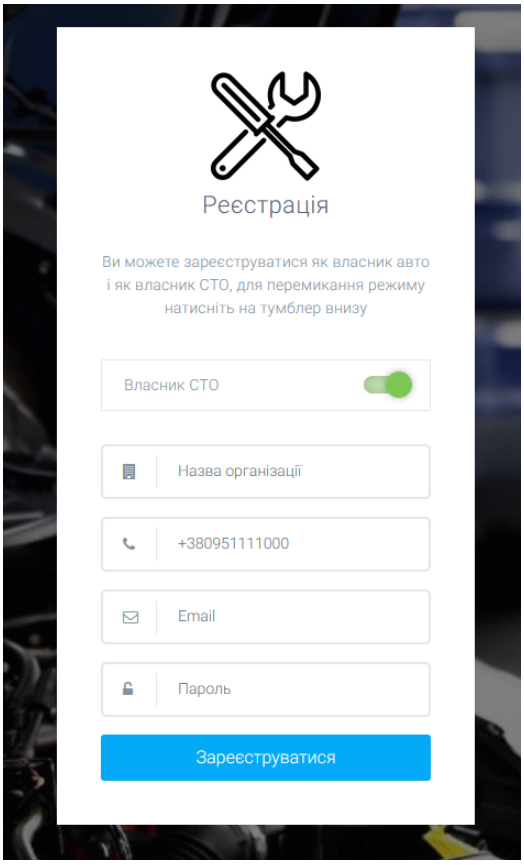


Рисунок 4.19 – Форма реєстрації організації

Після реєстрації користувач має можливість увійти в систему. Форма входу продемонстрована на рисунку 4.20.

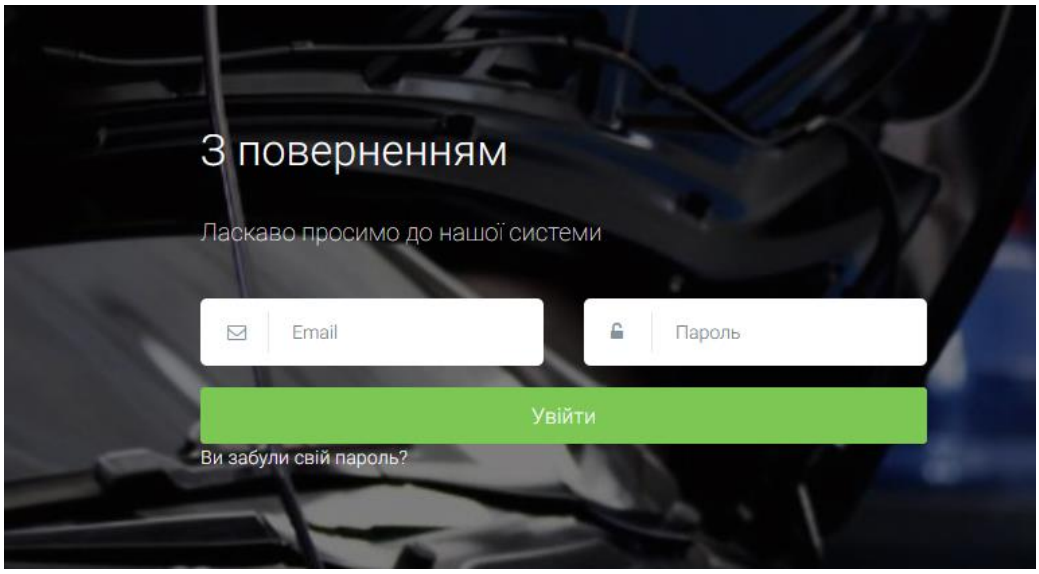


Рисунок 4.20 – Форма входу

Якщо користувач не пам’ятає пароль, існує можливість його відновлення за допомогою електронної пошти. Спочатку необхідно перейти на відповідне посилання на формі входу. Буде відкрита форма відновлення паролю (рис. 4.21), в якій необхідно ввести адрес власної електронної пошти.

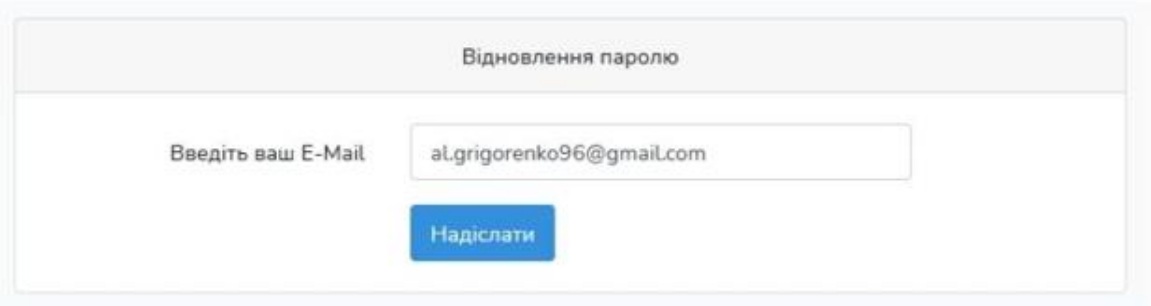


Рисунок 4.21 – Форма відновлення паролю

Система перевірить чи зареєстрований даний користувач. У разі підтвердження інформації, на адрес указаної електронної пошти буде надіслано відповідний лист (рис. 4.22) із посиланням на форму вводу нового пароля (рис. 4.23). Посилання у листу має певний термін дії – це 60 хвилин. По закінченню даного терміну посилання буде недійсним.

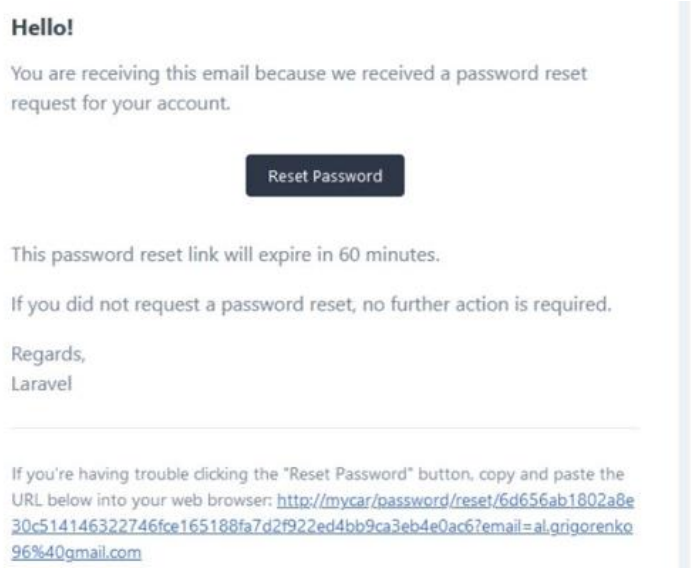


Рисунок 4.22 – Лист з посиланням

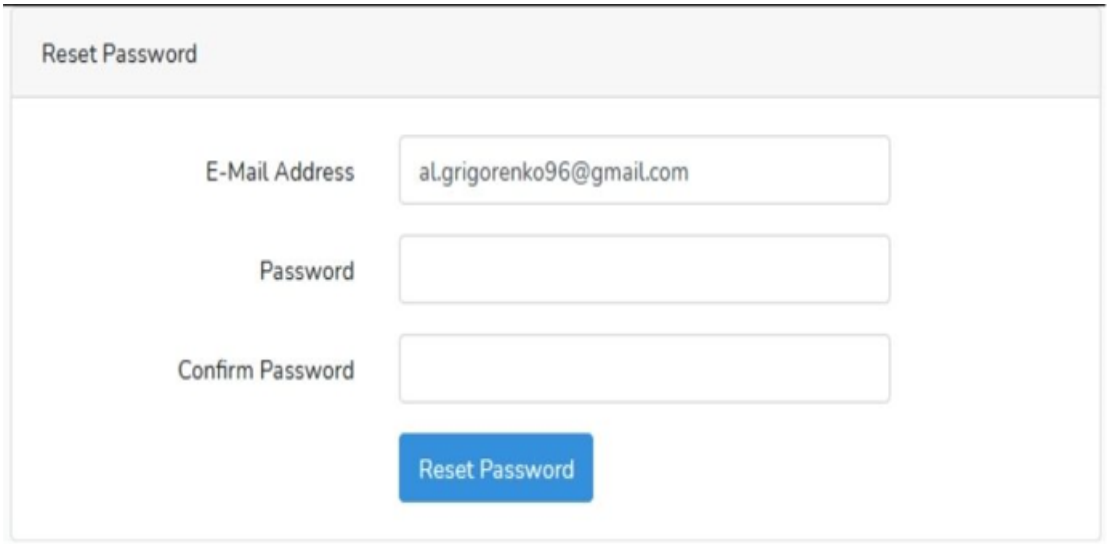


Рисунок 4.23 – Форма вводу нового пароля

При успішній авторизації система перевіряє тип користувача. У системі реалізовано два особистих кабінета. Окремо для клієнта та компанії.

При авторизація в залежності від типу користувача буде відкритий той чи інший кабінет. Інтерфейс кабінету клієнта та компанії відповідно продемонстровано на рисунках 4.24 – 4.25.

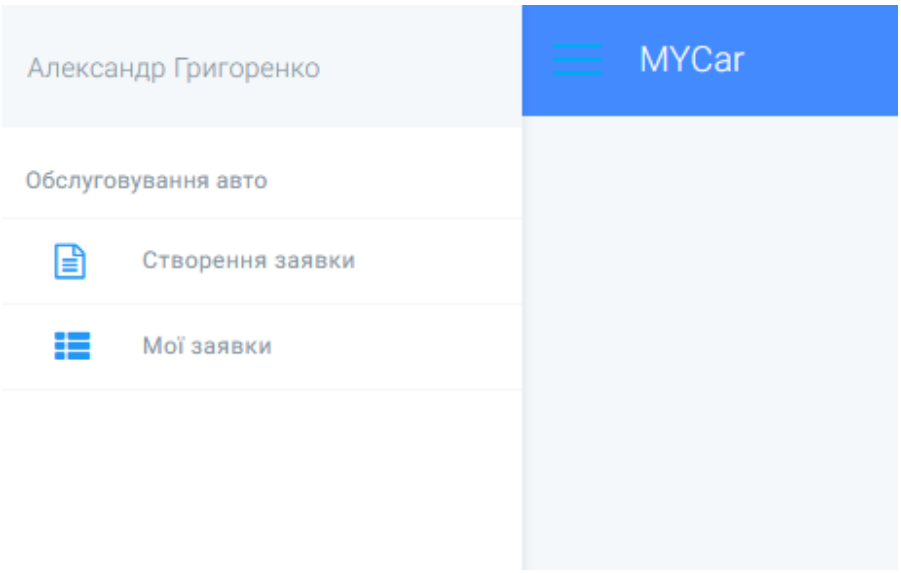


Рисунок 4.24 – Інтерфейс кабінету клієнта

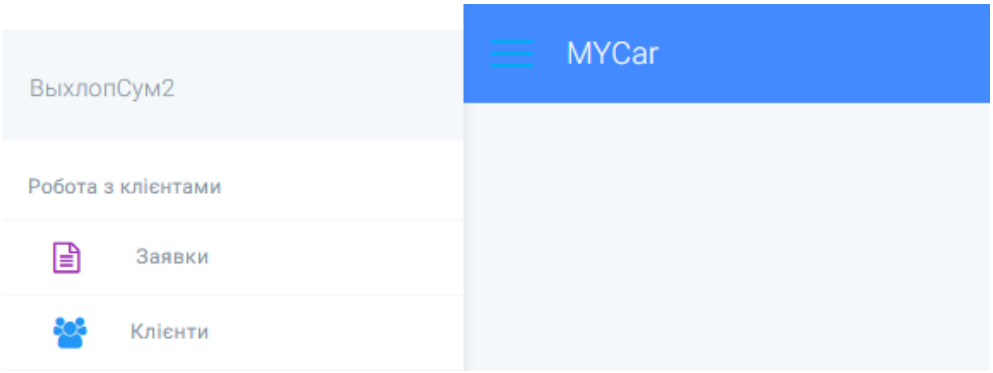


Рисунок 4.25 – Інтерфейс кабінету компанії

Також у системі є можливість редагування особистих даних як для клієнта, так і для компанії. Відповідні форми продемонстровані на рисунках 4.26–4.27.

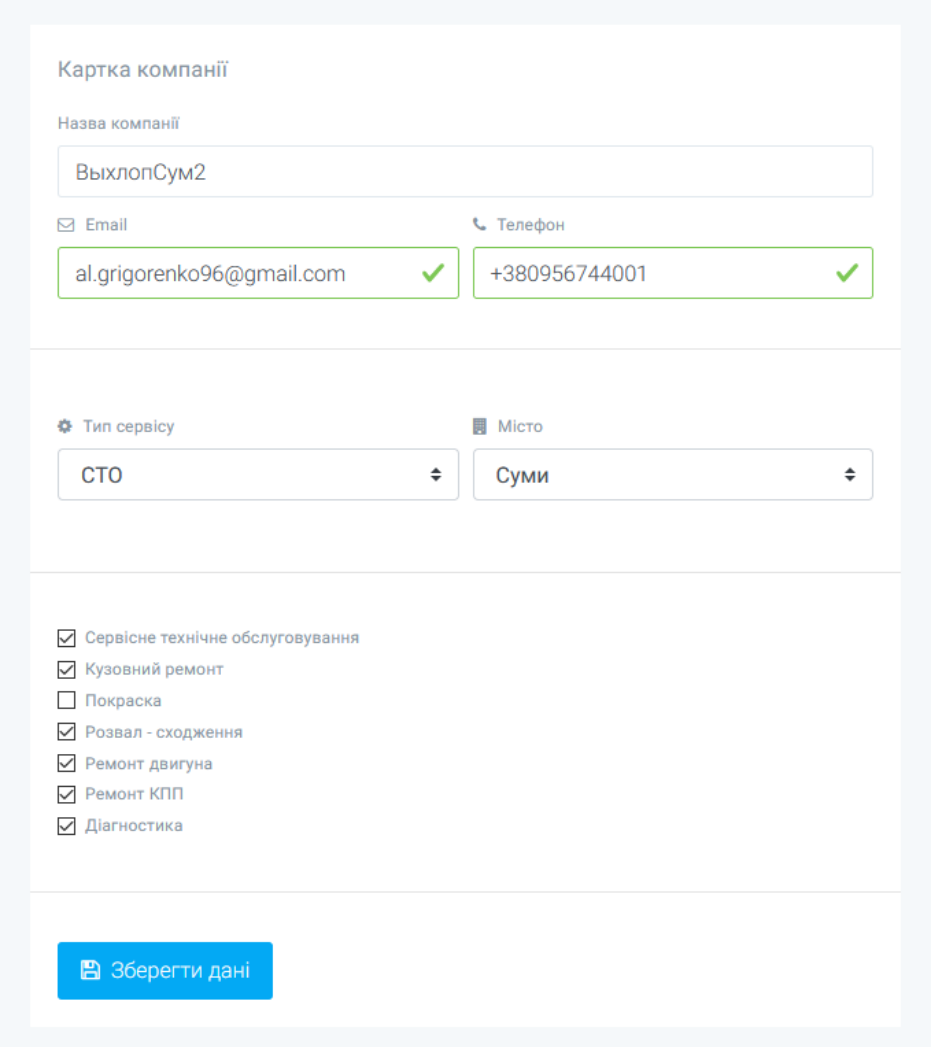


Рисунок 4.26 – Картка компанії

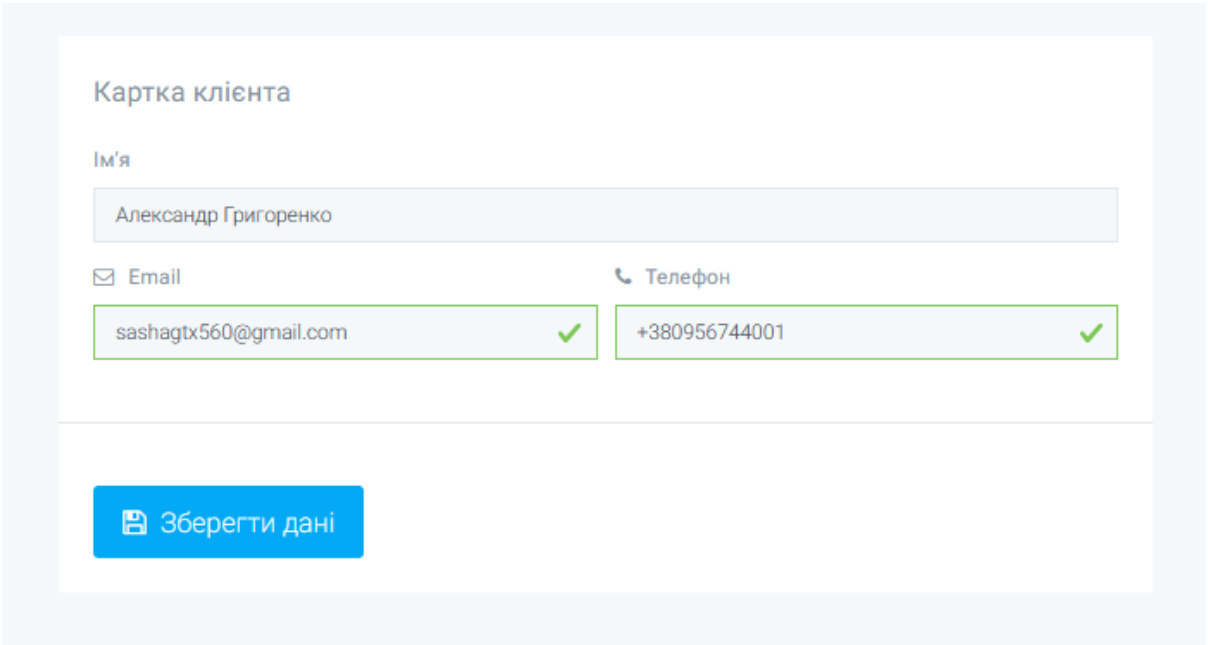


Рисунок 4.27 – Картка клієнта

Для компанії існує спеціальна перевірка, якщо деякі дані не заповнені – стане активним спеціальне повідомлення (рис. 4.28).

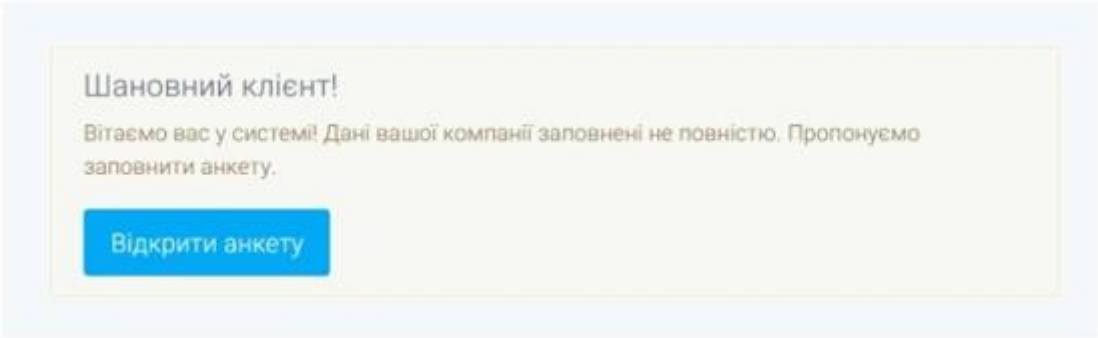


Рисунок 4.28 – Повідомлення про недостатньо заповнені дані

Кожен клієнт має можливість заповнити заявку на ремонт або діагностику власного транспортного засобу. Форма створення заявки показана рисунку 4.29.

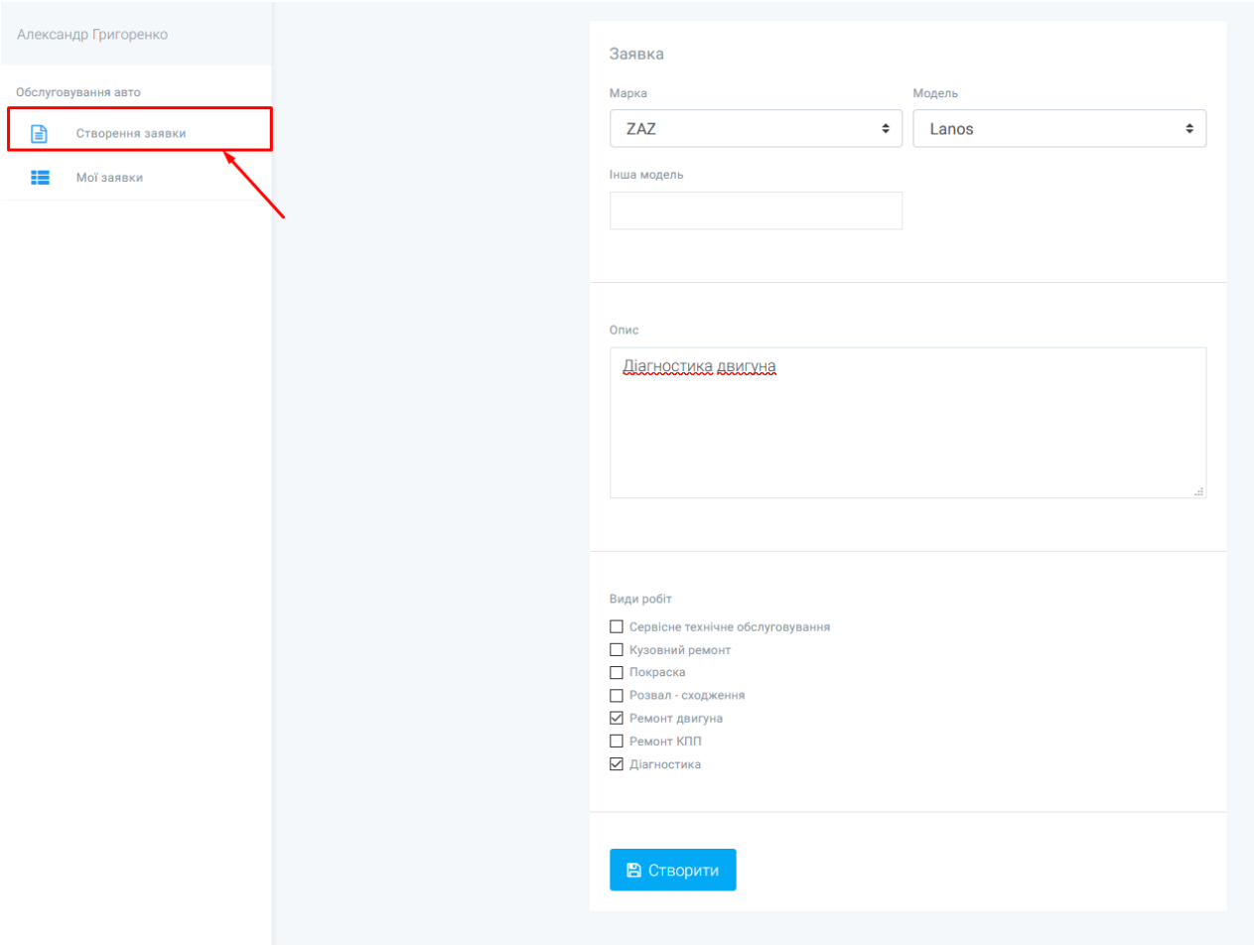


Рисунок 4.29 – Форма створення заявки

Також є можливість перегляду власних заявок та перегляду пропозицій від компаній щодо них. Інтерфейс наведено на рисунку 4.30.

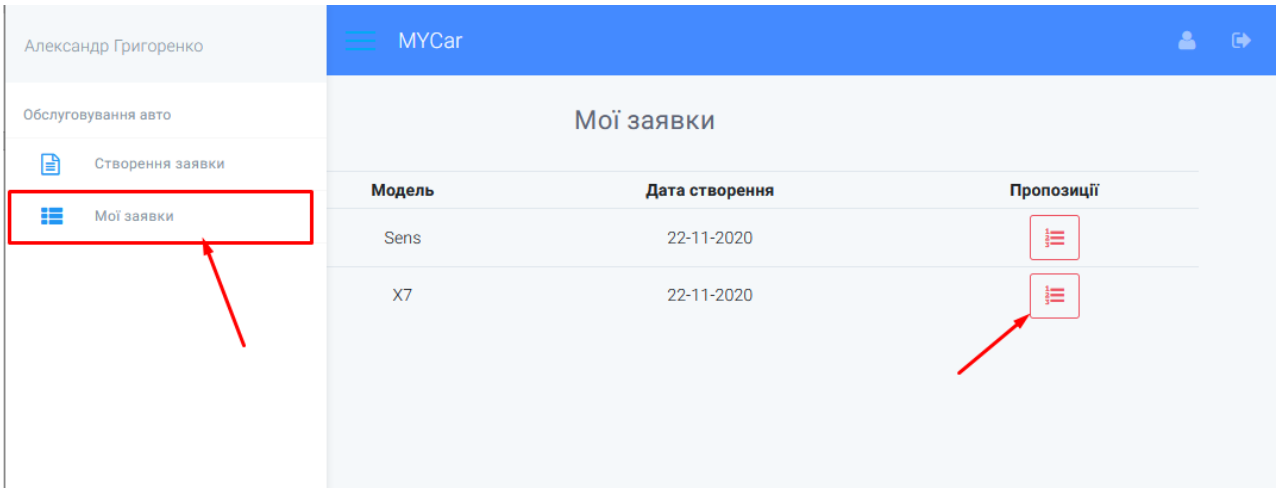


Рисунок 4.30 – Перегляд заявок

У свою чергу компанія має можливість переглядати заявки, які стосуються їхнього типу робіт. Якщо заявка цікава для них – є можливість відповісти на неї (рис. 4.31), оцінивши орієнтовану вартість робіт, час виконання та створення коментаря за необхідності. Форма відповіді зображена на рисунку 4.32.

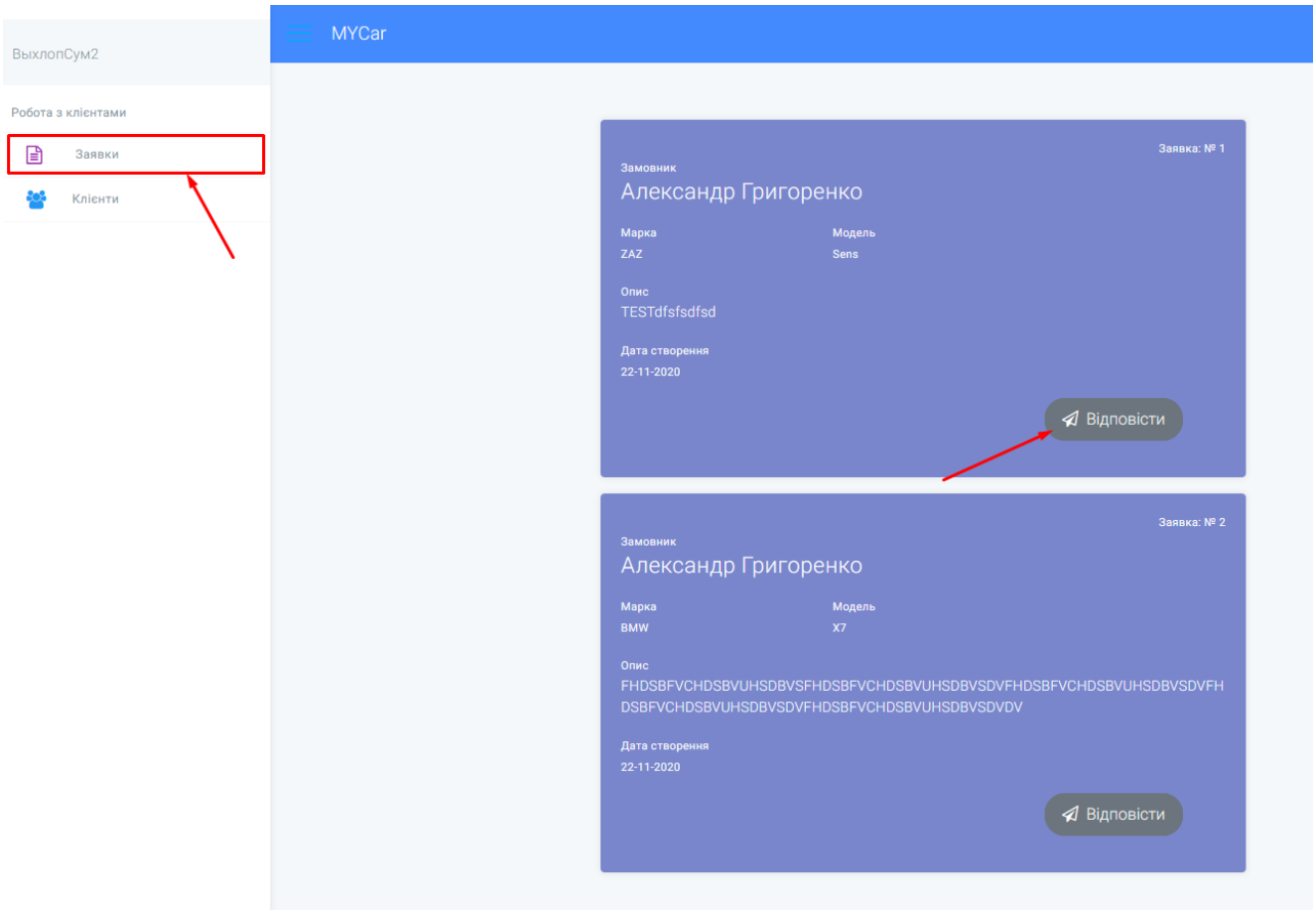


Рисунок 4.31 – Перегляд заявок компанією

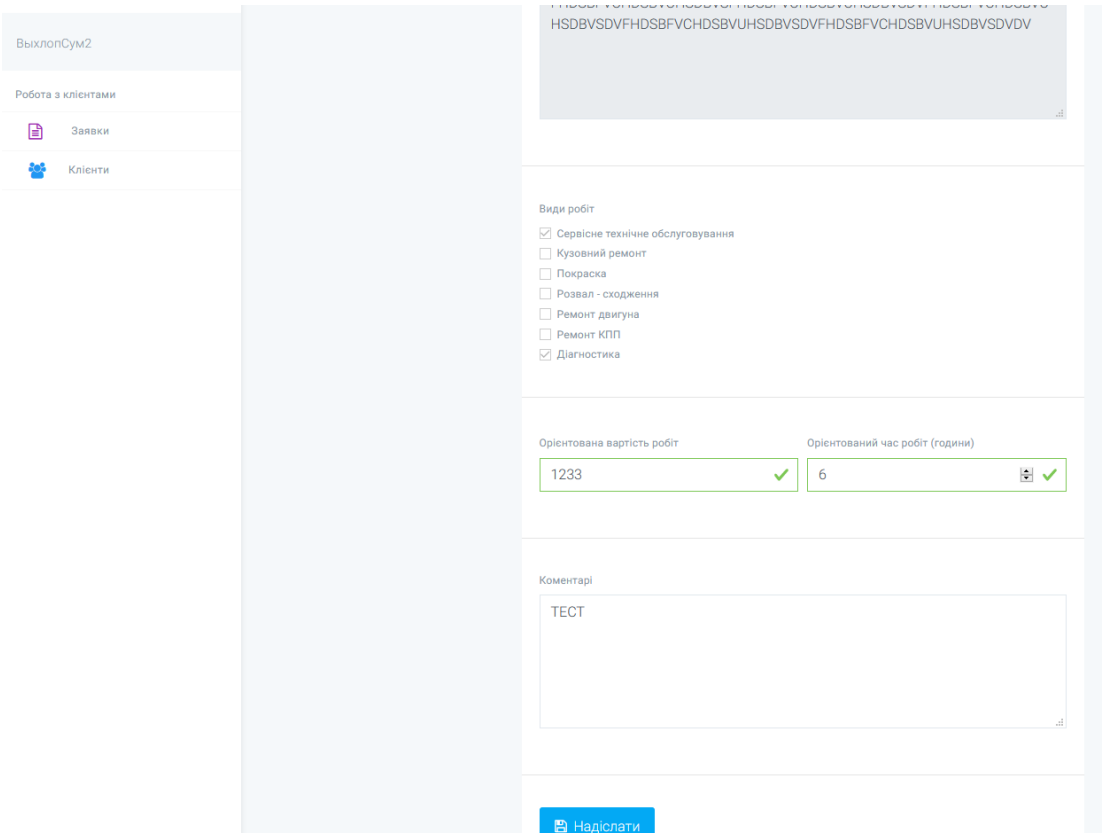


Рисунок 4.32 – Відповідь на заявку клієнта

При перегляді пропозицій для кожної заявки система за допомогою внутрішнього алгоритму шукає та надає найкращий варіант щодо вибору СТО.

Дана пропозиція буде з відповідним надписом «Вибір системи» та знаходитися в самому вверху (рис. 4.33). Це допоможе користувачу швидше обрати для себе найвигіднішу пропозицію.

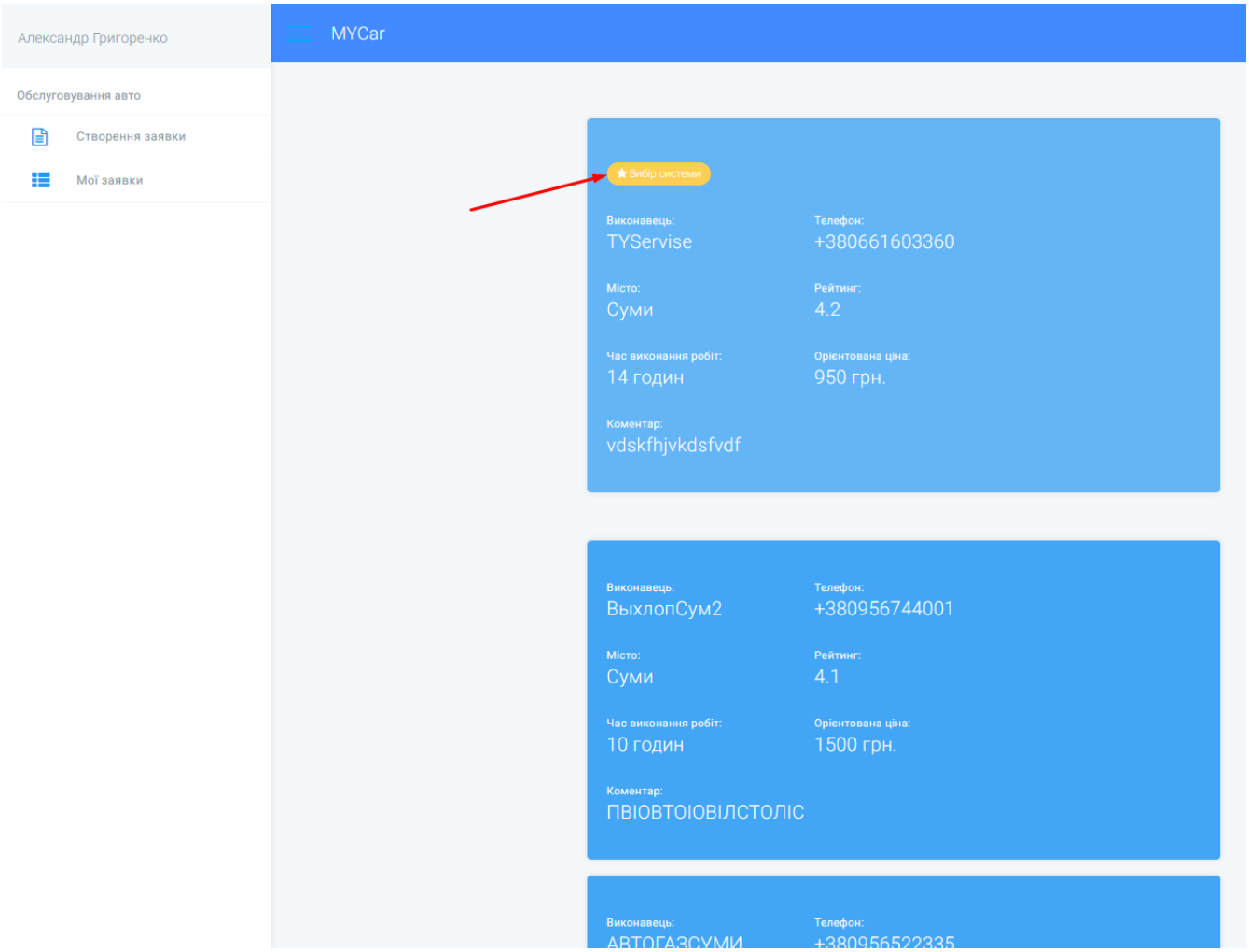


Рисунок 4.33 – Перелік пропозицій

У системі реалізована панель адміністрування (рис. 4.34), користувач із правом адміністратора має можливість переглядати та видаляти користувачів системи.

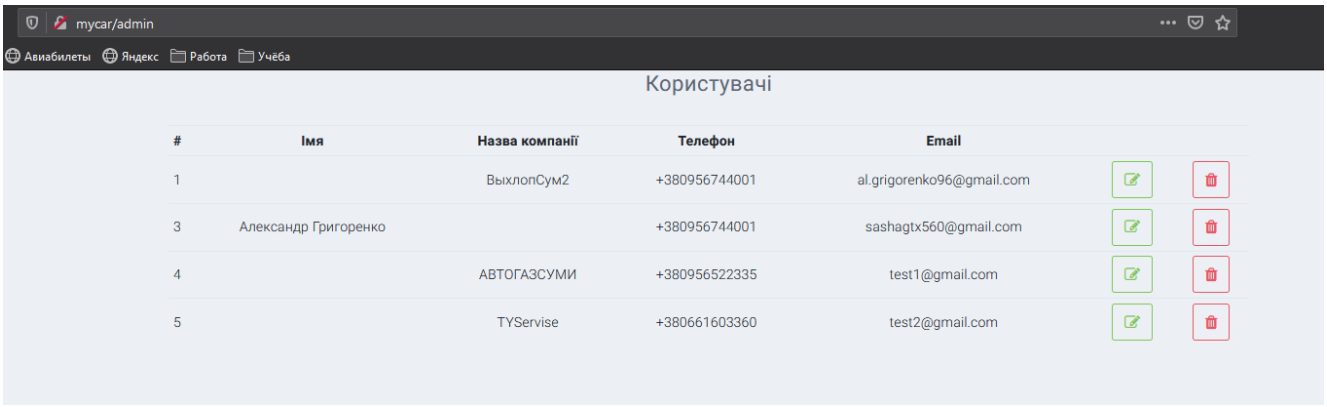


Рисунок 4.34 – Перелік користувачів

ВИСНОВОК

Враховуючи тенденції поширення автомобільного транспорту у світі і, зокрема, в Україні та велику кількість різноманітних станцій технічного обслуговування необхідно приділяти особливу увагу взаємодії між клієнтом, тобто автовласником авто, та СТО для їх більш продуктивної комунікації.

На ринку існують безліч інформаційних систем для пошуку СТО, але таких, які дозволяють клієнту, заповнивши лише декілька простих форм, отримати оптимальну для його потреб майстерню є досить мало.

Оскільки, даний ринок послуг є досить перспективним, враховуючи значне зростання кількості авто протягом декількох років, було вирішено створити інформаційну систему комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО для пошуку оптимального рішення за певними критеріями користувача. Визначено мету та задачі проекту.

Використання даної розробки дозволить автовласнику скоротити час та зусилля щодо вибору потрібної СТО з представлених системою варіантів за рахунок автоматизованого пошуку оптимального рішення на основі певних вимог та критеріїв. А також забезпечить належну організацію продуктивної взаємодії автовласників та станцій технічного обслуговування.

У подальшому дану інформаційну систему можна розширювати додатковими модулями для покриття більшої потреби клієнтів.

СПИСОК ЛІТЕРАТУРИ

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: 24.10.2020).

2. Distributorships and Dealerships URL: https://www.inc.com/encyclopedia/distributorships-and-dealerships.html (дата звернення: 24.10.2020).

3. Типи СТО URL: https://vse-sto.net/kak-pravilno-vybrat-avtoservis/ (дата звернення: 24.10.2020).

4. Як обрати СТО? URL: https://www.autocentre.ua/opyt/avtoservis/kak-vybrat-sto-5-osnovnyh-pravil-367236.html (дата звернення: 24.10.2020).

5. Прокин А.А., Богатырская В.А. ПЕРСПЕКТИВЫ РАЗВИТИЯ ИНТЕРАКТИВНЫХ WEB-САЙТОВ (дата звернення: 24.10.2020).

6. Cisco White Paper. The Zettabyte Era: Trends and Analysis. 2017. Available online: URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-indexvni/vni-hyperconnectivity-wp.pdf (дата звернення: 25.10.2020).

7. Pappas, S. Live Science. How Big Is the Internet, Really? Available online: http://www.livescience.com/54094-how-big-is-the-internet.html (дата звернення: 25.10.2020).

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: 25.10.2020).

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: 25.10.2020).

10. ESA Sentinel Online. Available online: https://sentinel.esa.int/web/sentinel/sentinel-data-access (accessed on 7 July 2017). (дата звернення: 25.10.2020).

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: 25.10.2020).

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: 25.10.2020).

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: 25.10.2020).

14. Why, when, and how to use the Google Map API URL: https://medium.com/@helennnsays/why-when-and-how-to-use-the-google-map-api-f5dfa35986dc (дата звернення: 25.10.2020).

15. Локальний сервер. URL: https://ospanel.io/. (дата звернення: 24.10.2020).

16. Andi Gutmans, Stig Bakken, Derick Rethans. PHP5 Power Programming. – Prentice Hall. – 704 с. (дата звернення: 24.10.2020).

17. Laravel – Функції безпеки. URL: https://webformyself.com/laravel-funkcii-bezopasnosti/. (дата звернення: 25.10.2020).

18. Никсон, Р. Создаем динамические web-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. - Москва 2016. URL: https://booster.by/files/oeu.pdf. (дата звернення: 26.10.2020).

19. Що таке CSS. URL: http://phpist.com.ua/css/5-whatcss. (дата звернення: 26.10.2020).

20. Хеслоп П. HTML самого початку. С.-Пб: Санкт-Петербург, 2014. . URL: http://ua-referat.com/Мова_розмітки_гіпертексту_-_HTML (дата звернення: 24.10.2020).

21. Переваги використання Laravel в IT URL: https://wezom.com.ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii (дата звернення: 26.10.2020).

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: 24.10.2020).

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: 24.10.2020).

24. Vue.JS: особливості, переваги та недоліки URL: https://jetruby.com/ru/blog/vue-js-preimuschestva-i-nedostatki/ (дата звернення: 26.10.2020).

25. Vue.js: особенности, применение и отличия от других Javascript фреймворков URL: https://stfalcon.com/ru/blog/post/vue-js-guide-to-tech (дата звернення: 26.10.2020).

26. Why PHP Storm? URL: https://www.jetbrains.com/phpstorm/ (дата звернення: 26.10.2020).

27. Продуктивне використання PHPStorm URL: https://habr.com/ru/post/157409/ (дата звернення: 26.10.2020).

28. An introduction to Git. URL: https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/. (дата звернення: 24.10.2020).

29. Системы управления версиями git и svn URL: https://hyperhost.ua/info/ru/sistemyi-upravleniya-versiyami-git-i-svn (дата звернення: 26.10.2020).

30. Git - Особенности, преимущества и недостатки URL: http://chinapads.ru/c/s/git_-_osobennosti_preimuschestva_i_nedostatki (дата звернення: 26.10.2020).

ДОДАТОК А ПЛАНУВАННЯ ІТ-ПРОЕКТУ

А.1 Ідентифікація мети проекту методом SMART

Визначення мети проекту буде проводитися 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 1.1

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Розробити веб-орієнтовану інформаційну 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 веб-орієнтовану інформаційну систему на основі реальних функціональних вимог
Relevant (реалістична)	Створити веб-орієнтовану інформаційну систему на основі сучасних фреймворків та шаблонів дизайну
Time-framed (обмежена у часі)	Створити веб-орієнтовану інформаційну 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.2.

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.3.

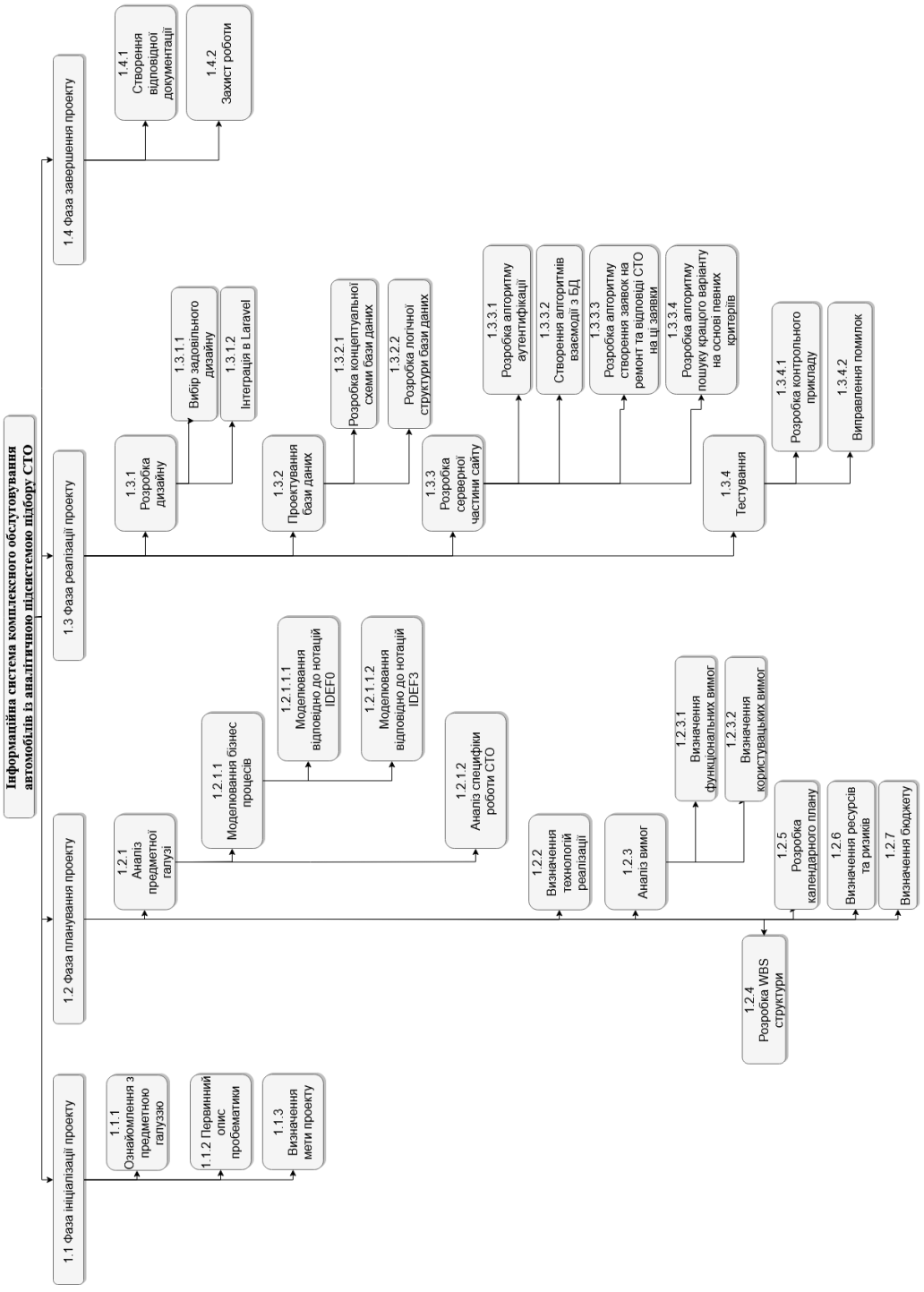


Рисунок А.2 – Діаграма декомпозиції робіт (WBS)

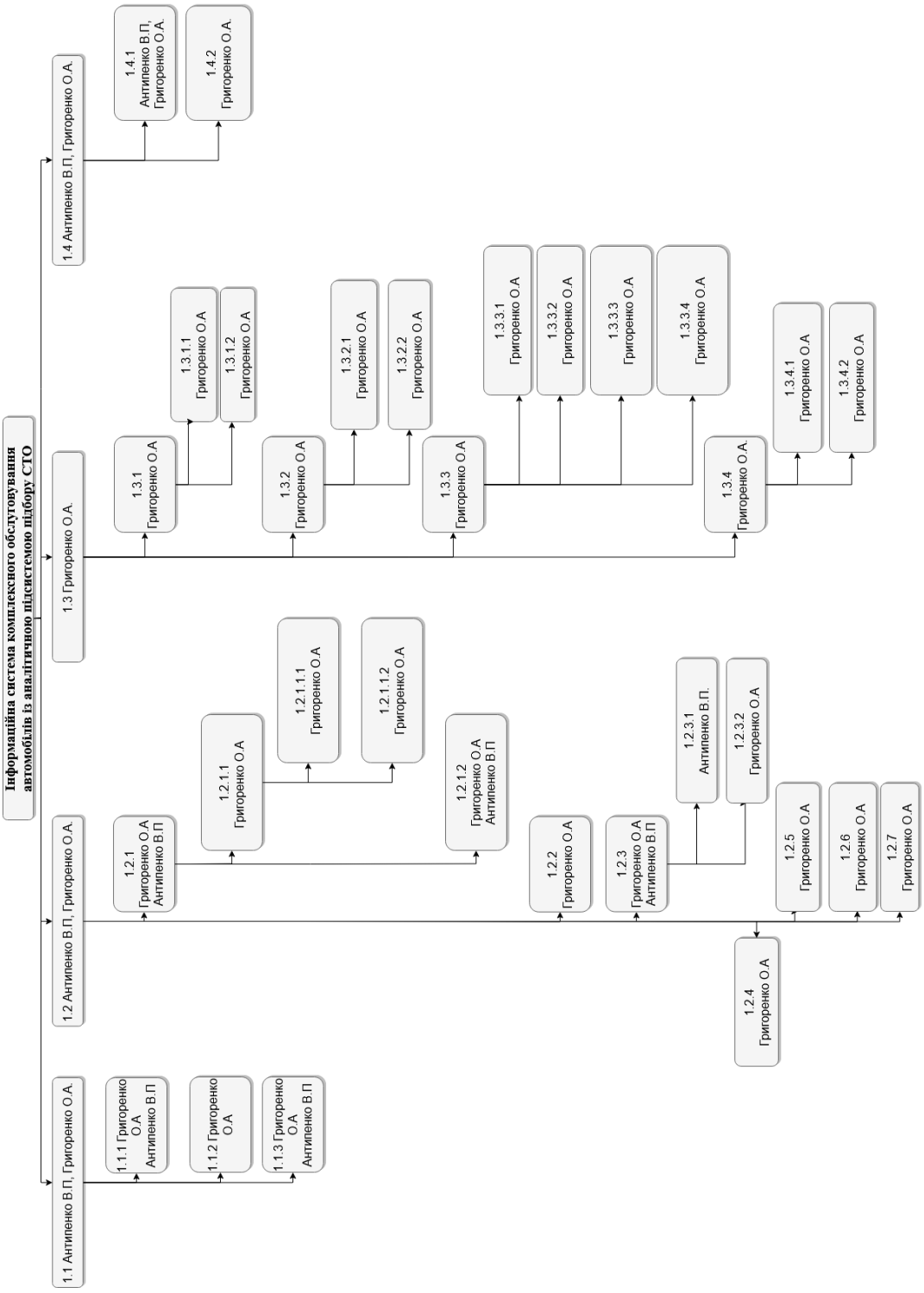


Рисунок А.3 – Діаграма організації робіт (OBS)

На основі даних із WBS та OBS діаграм будується матриця відповідальності. Вона потрібна для закріплення за кожною роботою зацікавлену сторону. Матриця відповідальності наведена у таблиці А.2.

Таблиця А.2 – Матриця відповідальності

Роль	Виконавець	Керівник
	Григоренко О.А	Антипенко В.П.
1. Створення інформаційної системи комплексного обслуговування автомобілів із аналітичною підсистемою підбору СТО	+	+
1.1 Фаза ініціалізації проекту	+	+
1.1.1 Ознайомлення з предметною галуззю	+	+
1.1.2 Первинний опис проблематики	+	
1.1.3 Визначення мети проекту	+	+
1.2 Фаза планування проекту	+	+
1.2.1 Аналіз предметної галузі	+	+
1.2.1.1 Моделювання бізнес процесів	+	
1.2.1.1.1 Моделювання IDEF0	+	
1.2.1.1.2 Моделювання IDEF3	+	
1.2.1.2 Аналіз специфіки роботи СТО	+	+
1.2.2. Визначення технологій реалізації	+	
1.2.3. Аналіз вимог	+	+
1.2.3.1. Визначення функціональних вимог		+
1.2.3.2. Визначення користувацьких вимог	+	
1.2.4 Розробка календарн плану	+	
1.2.5 Визначення ресурсів та ризиків	+	

Продовження таблиці А.2 – Матриця відповідальності

Роль	Виконавець	Керівник
	Григоренко О.А	Антипенко В.П.
1.2.5 Визначення бюджету	+	
1.2 Фаза реалізації проекту	+	
1.2.1 Розробка дизайну	+	
1.2.1.1 Вибір задовільного дизайну	+	
1.2.1.1 Інтеграція в Laravel	+	
1.2.2 Проектування БД	+	
1.2.2.1 Розробка концептуальної схеми	+	
1.2.2.2 Розробка логічної структури	+	
1.2.3 Розробка серверної частини	+	
1.2.3.1 Розробка алгоритму автентифікації	+	
1.2.3.2 Створення алгоритмів взаємодії з БД	+	
1.2.3.3 Розробка алгоритму створення заявок на ремонт	+	
1.2.3.4 Розробка алгоритму пошуку кращого варіанту на основі певних критеріїв	+	
1.2.4 Тестування	+	
1.2.4.1 Розробка контрольного прикладу	+	
1.2.4.2 Виправлення помилок	+	
1.3 Фаза завершення проекту	+	+
1.3.1 Створення документації	+	+
1.3.1 Захист роботи	+	

А.3 Побудова календарного графіку

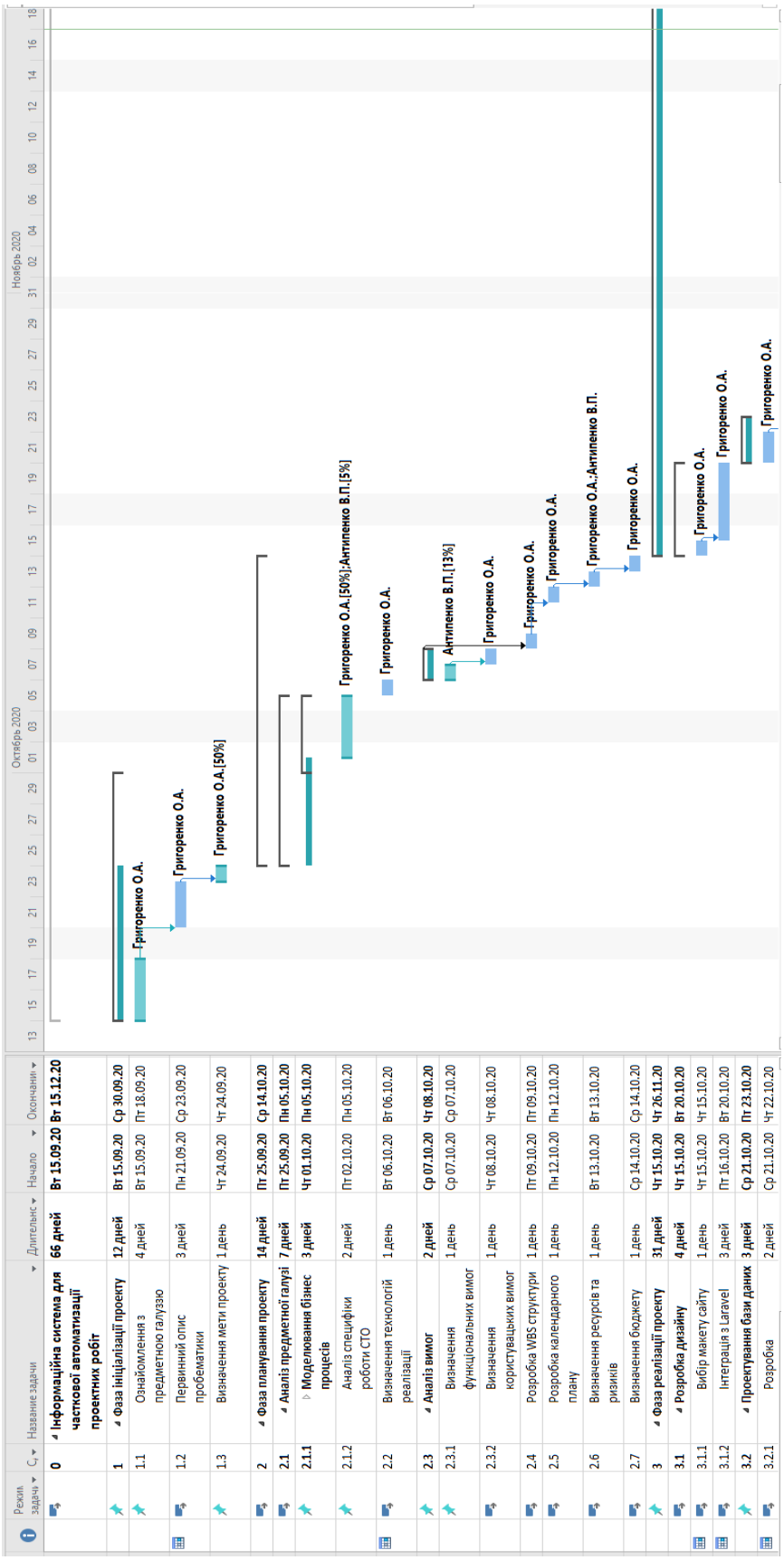
За допомогою таблиць досить незручно отримати уявлення про тривалість виконання робіт, враховувати часові обмеження.

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 А.3-А.4.



Режим задачи	С.	Название задачи	Длительнс	Начало	Окончани
	0	**Інформаційна система для часткової автоматизації проектних робіт**	**66 дней**	**Вт 15.09.20**	**Вт 15.12.20**
	1	**Фаза ініціалізації проекту**	**12 дней**	**Вт 15.09.20**	**Ср 30.09.20**
	1.1	Ознайомлення з предметною галузю	4 дней	Вт 15.09.20	Пт 18.09.20
	1.2	Первинний опис пробематики	3 дней	Пн 21.09.20	Ср 23.09.20
	1.3	Визначення мети проекту	1 день	Чт 24.09.20	Чт 24.09.20
	2	**Фаза планування проекту**	**14 дней**	**Пт 25.09.20**	**Ср 14.10.20**
	2.1	**Аналіз предметної галузі**	**7 дней**	**Пт 25.09.20**	**Пн 05.10.20**
	2.1.1	**Моделювання бізнес процесів**	**3 дней**	**Чт 01.10.20**	**Пн 05.10.20**
	2.1.2	Аналіз специфіки роботи СТО	2 дней	Пт 02.10.20	Пн 05.10.20
	2.2	Визначення технологій реалізації	1 день	Вт 06.10.20	Вт 06.10.20
	2.3	**Аналіз вимог**	**2 дней**	**Ср 07.10.20**	**Чт 08.10.20**
	2.3.1	Визначення функціональних вимог	1 день	Ср 07.10.20	Ср 07.10.20
	2.3.2	Визначення користувацьких вимог	1 день	Чт 08.10.20	Чт 08.10.20
	2.4	Розробка WBS структури	1 день	Пт 09.10.20	Пт 09.10.20
	2.5	Розробка календарного плану	1 день	Пн 12.10.20	Пн 12.10.20
	2.6	Визначення ресурсів та ризиків	1 день	Вт 13.10.20	Вт 13.10.20
	2.7	Визначення бюджету	1 день	Ср 14.10.20	Ср 14.10.20
	3	**Фаза реалізації проекту**	**31 дней**	**Чт 15.10.20**	**Чт 26.11.20**
	3.1	**Розробка дизайну**	**4 дней**	**Чт 15.10.20**	**Вт 20.10.20**
	3.1.1	Вибір макету сайту	1 день	Чт 15.10.20	Чт 15.10.20
	3.1.2	Інтеграція з Laravel	3 дней	Пт 16.10.20	Вт 20.10.20
	3.2	**Проектування бази даних**	**3 дней**	**Ср 21.10.20**	**Пт 23.10.20**
	3.2.1	Розробка	2 дней	Ср 21.10.20	Чт 22.10.20

Рисунок А.3 – Діаграма Ганта (блок 1)

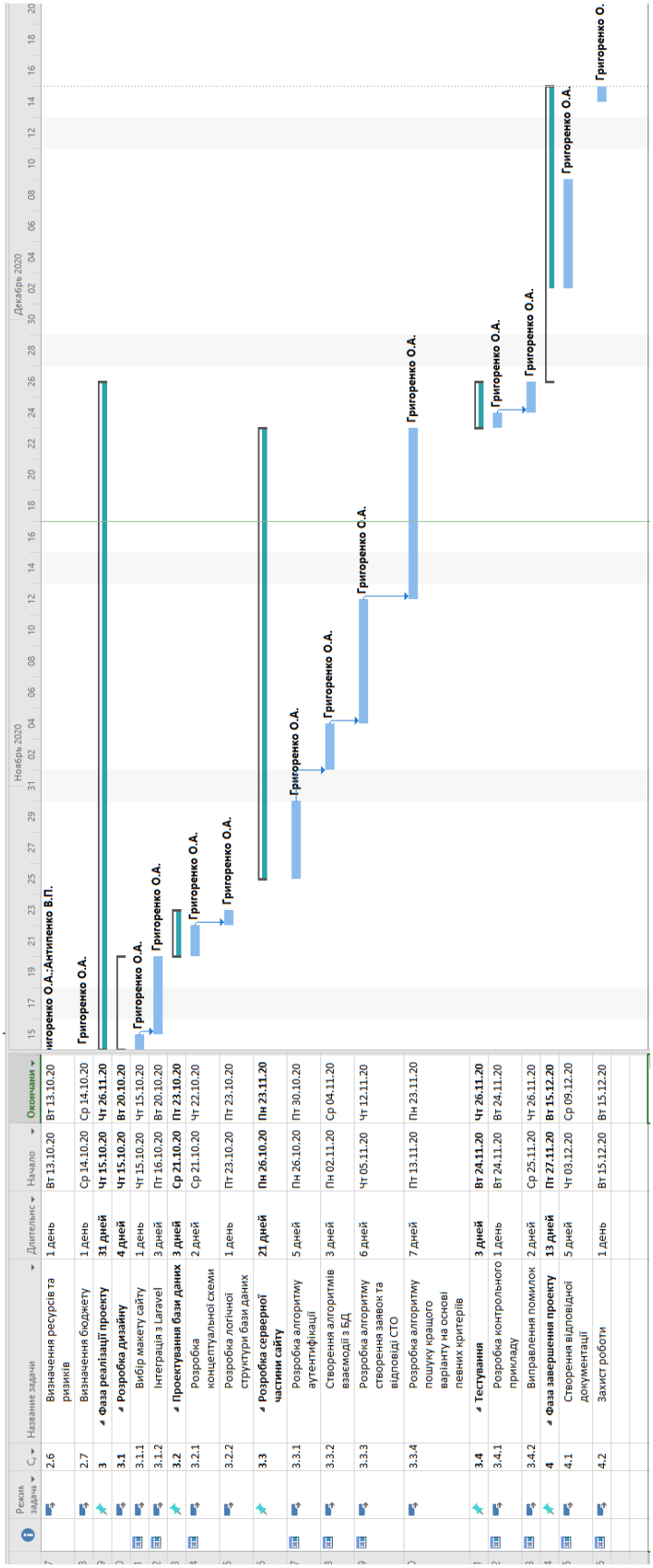


Рисунок А.4 – Діаграма Ганта (блок 2)

А.4 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 1.3.

Таблиця А.3 – 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 А.5.

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**

Втрати

Рисунок А.5 – 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 Б ПРОГРАМНИЙ КОД

Б.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');
        }
    }

}
```

Б.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(),
        ]);

    }

}
```

Б.3 ApplicationController

```
<?php

namespace App\Http\Controllers;

use App\Application;
use App\Brand;
use App\City;
use App\CompanyInfo;
use App\Models;
use App\Response;
use App\TypeService;
use App\TypeWork;
use App\TypeWorkForApplication;
use App\TypeWorkForCompany;
use App\User;
use Illuminate\Http\Request;

class ApplicationController 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()
    {
        $applications = Application::select(
            'users.name as name_client',
            'application.id',
            'application.description',
            'application.created_at',
            'brand.name AS name_brand',
            'model.name AS name_model'
        )
            ->leftJoin('users', 'users.id', '=', 'application.id_client')
            ->leftJoin('brand', 'brand.id', '=', 'application.id_brand')
            ->leftJoin('model', 'model.id', '=', 'application.id_model')
```

```
        ->get();

    return view('applications.index',
        [
            'applications' => $applications,
        ]);
}

public function myApplication()
{
    $my_applications = Application::select(
        'users.name as name_client',
        'application.id',
        'application.description',
        'application.created_at',
        'brand.name AS name_brand',
        'model.name AS name_model'
    )
        ->where('id_client',\Auth::user()->id)
        ->leftJoin('users', 'users.id', '=', 'application.id_client')
        ->leftJoin('brand', 'brand.id', '=', 'application.id_brand')
        ->leftJoin('model', 'model.id', '=', 'application.id_model')
        ->get();

    return view('applications.my_application',
        [
            'applications' => $my_applications,
        ]);
}

public function create()
{
    $user_info =\Auth::user();

    return view('applications.create',
        [
            'brands' =>  Brand::select('id','name')->get()->toArray(),
            'type_works' => TypeWork::select('id','name')->get()->toArray(),
            'id_client' =>  $user_info->id,

        ]);
}

public function store(Request $request)
{
    $new_application = new Application([
        'id_client' => $request->id_client,
        'description' => $request->description,
        'id_brand' => $request->brand,
        'id_model' => $request->model,
    ]);

    $new_application->save();

    $last_application = Application::latest()->first();

    foreach ($request->checked_works as $key => $value) {
        $type_work = new TypeWorkForApplication([
            'id_application' => $last_application->id,
            'id_type_work' => $value,
        ]);
        $type_work->save();
    }
}
```

```
    public function storeResponse(Request $request)
    {
        $response = new Response([
            'id_application' => $request->id_application,
            'id_client' => $request->id_client,
            'id_company' => \Auth::user()->id,
            'price' => $request->price,
            'time' => $request->time,
            'comment' => $request->comment,
        ]);

        $response->save();
    }

    public function getSelectModel($id_brand)
    {
            $models = Models::select('id','name')->where('id_brand', $id_brand)-
>get()->toArray();

            return $models;
    }

    public function responseIndexForApp($id_application)
    {
        return redirect()->action(
            [ApplicationController::class, 'responseIndexForAppSort'],
['id_application' => $id_application]
        );

    }

    public function responseIndexForAppSort(Request $request){

        $min_price_response = Response::select(
            'company_info.rating as rating',
            'users.name_company as name_company',
            'users.phone as phone',
            'cities.name as city',
            'responses.id',
            'responses.id_application',
            'responses.id_company',
            'responses.price',
            'responses.time',
            'responses.comment'
        )
            ->where('id_application',$request->id_application)
            ->leftJoin('company_info', 'company_info.id_company', '=',
'responses.id_company')
            ->leftJoin('cities', 'cities.id', '=', 'company_info.id_city')
            ->leftJoin('users', 'users.id', '=', 'responses.id_company')
            ->orderBy('price', 'asc')
            ->first();

        $min_time_response = Response::select(
            'company_info.rating as rating',
            'users.name_company as name_company',
            'users.phone as phone',
            'cities.name as city',
            'responses.id',
            'responses.id_application',
            'responses.id_company',
            'responses.price',
            'responses.time',
            'responses.comment'
        )
```

```
            ->where('id_application',$request->id_application)
            ->leftJoin('company_info', 'company_info.id_company', '=',
'responses.id_company')
            ->leftJoin('cities', 'cities.id', '=', 'company_info.id_city')
            ->leftJoin('users', 'users.id', '=', 'responses.id_company')
            ->orderBy('time', 'asc')
            ->first();

        $max_rating_response = Response::select(
            'company_info.rating as rating',
            'users.name_company as name_company',
            'users.phone as phone',
            'cities.name as city',
            'responses.id',
            'responses.id_application',
            'responses.id_company',
            'responses.price',
            'responses.time',
            'responses.comment'
        )
            ->where('id_application',$request->id_application)
            ->leftJoin('company_info', 'company_info.id_company', '=',
'responses.id_company')
            ->leftJoin('cities', 'cities.id', '=', 'company_info.id_city')
            ->leftJoin('users', 'users.id', '=', 'responses.id_company')
            ->orderBy('rating', 'desc')
            ->first();

        $min_price_response['weight'] = '0.4';
        $min_time_response['weight'] = '0.35';
        $max_rating_response['weight'] = '0.25';

        if ($min_time_response->id == $max_rating_response->id){
            $best_response = $max_rating_response;
        }
        else{
            $best_response = $min_price_response;
        }

        $rest_responses = Response::select(
            'company_info.rating as rating',
            'users.name_company as name_company',
            'users.phone as phone',
            'cities.name as city',
            'responses.id',
            'responses.id_application',
            'responses.id_company',
            'responses.price',
            'responses.time',
            'responses.comment'
        )
            ->where('id_application',$request->id_application)
            ->where('responses.id','!=',$best_response->id)
            ->leftJoin('company_info', 'company_info.id_company', '=',
'responses.id_company')
            ->leftJoin('cities', 'cities.id', '=', 'company_info.id_city')
            ->leftJoin('users', 'users.id', '=', 'responses.id_company')
            ->get()
            ->toArray();

        return view('applications.responses_for_app',
            [
                'best_response' => $best_response,
                'rest_responses' => $rest_responses,
            ]);
```

```
    }

    public function responseCreate($id_application)
    {
        return redirect()->action(
            [ApplicationController::class, 'responseIndex'], ['id_application' =>
$id_application]
        );

    }
    public function responseIndex(Request $request){

        $application_info = Application::select(
            'users.name as name_client',
            'users.id as id_client',
            'application.id',
            'application.description',
            'application.created_at',
            'brand.name AS name_brand',
            'model.name AS name_model'
        )
            ->where('application.id',$request->id_application)
            ->leftJoin('users', 'users.id', '=', 'application.id_client')
            ->leftJoin('brand', 'brand.id', '=', 'application.id_brand')
            ->leftJoin('model', 'model.id', '=', 'application.id_model')
            ->first();

        $application_works = TypeWorkForApplication::where('id_application',
$request->id_application)->get()->toArray();

        return view('applications.response',
            [
                'application_info' => $application_info,
                'application_works' => $application_works,
                'type_works' => TypeWork::select('id','name')->get()->toArray(),
            ]);

    }


}
```

Б.4 ApplicationForm.vue

```
<template>
    <div class="container">
            <div class="row my-3">
                <div class="col-md-7  offset-md-2">
                    <form @submit.prevent="onSubmit" method="post">
                        <div class="card no-b  no-r">
                            <div class="card-body">
                                <h5 class="card-title">Заявка</h5>
                                <div class="form-row mt-1">
                                    <div class="form-group col-6 ">
                                        <label for="type_service" class="col-form-
label s-12">Марка</label>
                                        <select class="custom-select"
                                                id="type_service"
                                                v-model="form.brand">
                                            <option v-for="(option, key) in
optionsBrandList"
                                                    :value=option.id
```

```
                                        :key="key">
                                    {{ option.name }}
                                </option>

                            </select>
                        </div>

                        <div class="form-group col-6 ">
                            <label for="city" class="col-form-label s-
12">Модель</label>
                            <select class="custom-select"
                                    id="city"
                                    :disabled="!form.brand"
                                    v-model="form.model">
                                <option v-for="(option, key) in
optionsModelsList"
                                        :value=option.id
                                        :key="key">
                                    {{ option.name }}
                                </option>
                            </select>
                        </div>

                        <div class="form-group col-6 ">
                            <label for="time">Інша модель</label>
                            <input class="form-control r-0"
type="text"  id="time">
                        </div>
                    </div>
                </div>
                <hr>

                <div class="card-body">
                    <div class="form-group focused">
                        <label for="description">Опис</label>
                        <textarea class="form-control r-0"   v-
model="form.description" id="description" rows="5"></textarea>
                    </div>
                </div>
                <hr>

                <div class="card-body">
                    <label for="description">Види робіт</label>
                    <div class="form-check" v-for="(type_work, k) in
optionsTypeWorksList" :key="k">
                        <label class="form-check-label">
                            <input type="checkbox" class="form-check-
input"

:checked="form.checked_works.indexOf(+type_work.id)>-1"
                                   :value="type_work.id"
                                   v-
model="form.checked_works">{{type_work.name}}
                        </label>
                    </div>
                </div>

                <hr>

                <div class="card-body">
                    <button type="submit" class="btn btn-primary btn-
lg"><i class="fa fa-save mr-2"></i>Створити</button>
                    <div v-if="saveSuccess" role="alert" class="alert
alert-success" style ="margin-top: 20px" ><strong>Дані успішно
збережені!</strong></div>
```

```
                            </div>
                        </div>
                    </form>
                </div>
            </div>
    </div>
</template>

<script>
    import applicationApi from "../applications/application.api";
    import DatePicker from 'vue2-datepicker';
    import 'vue2-datepicker/index.css';

    export default {
        name: "ApplicationForm",
        components: { DatePicker },
        props: [
            'brands',
            'idClient',
            'typeWorks',
        ],

        data() {
            return {
                form: {
                    _method:'post',
                    id_client:'',
                    brand:'',
                    model: '',
                    description: '',
                    checked_works: []
                },

                optionsBrandList:[],
                optionsModelsList:[],
                optionsTypeWorksList:[],
                saveSuccess: false,
                csrf: document.querySelector('meta[name="csrf-
token"]').getAttribute('content'),
            }
        },

        mounted() {
            /**
             * Загрузка данных
             */

            this.form.id_client = this.idClient;
            this.optionsBrandList = this.brands;
            this.optionsTypeWorksList = this.typeWorks;

        },

        watch: {
            'form.brand': function (){
                applicationApi.getSelectModel(this.form.brand)
                    .then((response) => {
                        this.optionsModelsList = response.data;
                    });
            }
        },

        methods:{
            onSubmit(){
                applicationApi.storeApplication(this.form)
```

```
                    .then((response) => {
                    this.saveSuccess = true;
                });
                window.location.href = '/home';
            },
        }
    }
</script>

<style scoped>

</style>
```

Б.4 my_application.blade.php

```
@extends('layouts.main_client')

@section('content')
    <div class="container">

        <div class="box">
            <div class="box-header">
                <h3 class="box-title"><strong>Мої заявки</strong></h3>
            </div>
            <br>
            <!-- /.box-header -->
            <div class="box-body no-padding">
                <table class="table table-sm">
                    <tbody>
                    <tr>
                        <th style="width: 10px">#</th>
                        <th>Марка</th>
                        <th>Модель</th>
                        <th>Дата створення</th>
                        <th>Пропозиції</th>
                    </tr>
                    @foreach($applications as $application)
                    <tr>
                        <td>{{$application->id}}</td>
                        <td>{{$application->name_brand}}</td>
                        <td>{{$application->name_model}}</td>
                        <td>{{$application->created_at->format('d-m-Y')}}</td>
                        <td>
                            <form action="{{url('/application_responses',
[$application->id])}}" method="POST">
                                <input type="hidden" name="_token" value="{{
csrf_token() }}">
                                <button type="submit" class="btn btn-outline-
danger"><i class="fa fa-list-ol"></i></button>
                            </form>
                        </td>
                    </tr>
                    @endforeach
                    </tbody>
                </table>
            </div>
            <!-- /.box-body -->
        </div>
    </div>
@endsection

@extends('layouts.main_client')
```

```
@section('content')
    <div class="container">
        <div class="row my-3">
            <div class="col-md-7  offset-md-2">
                    <div class="card no-b my-3 shadow blue lighten-2 text-white">
                        <div class="card-body">

                            <div class="my-4">
                                <span class="badge badge-warning badge-pill"><i
class="fa fa-star"></i> Вибір системи</span>
                            </div>

                            <div class="my-3">
                                <div class="row">
                                    <div class="col-md-4">
                                        <small>Виконавець:</small>
                                        <h4>{{$best_response->name_company}}</h4>
                                    </div>
                                    <div class="col-md-4">
                                        <strong> <small>Телефон:</small></strong>
                                        <div> <h4>{{$best_response-
>phone}}</h4></div>
                                    </div>
                                </div>
                            </div>
                            <div class="my-3">
                                <div class="row">
                                    <div class="col-md-4">
                                        <strong> <small>Місто:</small></strong>
                                        <div><h4>{{$best_response-
>city}}</h4></div>
                                    </div>
                                    <div class="col-md-8">
                                        <strong> <small>Рейтинг:</small></strong>
                                        <div> <h4>{{$best_response-
>rating}}</h4></div>
                                    </div>
                                </div>
                            </div>

                            <div class="my-3">
                                <div class="row">
                                    <div class="col-md-4">
                                        <strong> <small>Час виконання
робіт:</small></strong>
                                        <div><h4>{{$best_response->time}}
годин</h4></div>
                                    </div>
                                    <div class="col-md-8">
                                        <strong> <small>Орієнтована
ціна:</small></strong>
                                        <div> <h4>{{$best_response->price}}
грн.</h4></div>
                                    </div>
                                </div>
                            </div>

                            <div class="my-3">
                                <small>Коментар:</small>
                                <h4>{{$best_response->comment}}</h4>
                            </div>
                        </div>
                    </div>
            </div>
        </div>
```

```
        <div class="row my-3">
            <div class="col-md-7  offset-md-2">
                @foreach($rest_responses as $response)

                    <div class="card no-b my-3 shadow blue lighten-1 text-white">
                        <div class="card-body">
                            <div class="my-3">
                                <div class="row">
                                    <div class="col-md-4">
                                        <small>Виконавець:</small>
                                        <h4>{{$response['name_company']}}</h4>
                                    </div>
                                    <div class="col-md-4">
                                        <strong> <small>Телефон:</small></strong>
                                        <div>
<h4>{{$response['phone']}}</h4></div>
                                    </div>
                                </div>
                            </div>
                            <div class="my-3">
                                <div class="row">
                                    <div class="col-md-4">
                                        <strong> <small>Місто:</small></strong>
                                        <div><h4>{{$response['city']}}</h4></div>
                                    </div>
                                    <div class="col-md-8">
                                        <strong> <small>Рейтинг:</small></strong>
                                        <div>
<h4>{{$response['rating']}}</h4></div>
                                    </div>
                                </div>

                                <div class="my-3">
                                    <div class="row">
                                        <div class="col-md-4">
                                            <strong> <small>Час виконання
робіт:</small></strong>
                                            <div><h4>{{$response['time']}}
годин</h4></div>
                                        </div>
                                        <div class="col-md-8">
                                            <strong> <small>Орієнтована
ціна:</small></strong>
                                            <div> <h4>{{$response['price']}}
грн.</h4></div>
                                        </div>
                                    </div>
                                </div>

                                <div class="my-3">
                                    <small>Коментар:</small>
                                    <h4>{{$response['comment']}}</h4>
                                </div>
                            </div>
                        </div>

                @endforeach
            </div>
        </div>

    </div>
@endsection
```

ДОДАТОК В АКТ ВПРОВАДЖЕННЯ

Товариство з обмеженою відповідальністю
«КАСІКО»

40016, м. Суми, вул. Харківська, 30/2, код ЄДРПОУ 41189239
рахунок № 26004055008989 в Сумській філії ПАТ КБ «Приватбанк» МФО 337546

АКТ

впровадження результатів дипломної роботи

студента Сумського державного університету

Григоренка Олександра Андрійовича

Даний акт підтверджує, що результати роботи студента Григоренко Олександра Андрійовича на тему «Веб-орієнтована інформаційна система комплексного обслуговування автомобілів із аналітичною підсистемою підбору станції технічного обслуговування» впроваджено для використання у подальшій розробці власного програмного забезпечення.

Надана інформаційна система дозволить автовласнику скоротити час та зусилля щодо вибору потрібної СТО з представлених системою варіантів за рахунок автоматизованого пошуку оптимального рішення.

Директор ТОВ «КАСІКО» Кас'ян Л.О.

"КАСІКО" Ідентифікаційний код 41189239 УКРАЇНА м.СУМИ

Рисунок В.1 – Акт впровадження