МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«___»_____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИЦІ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТРА»
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ «ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

Тема: «WEB–додаток для надання послуг особистого каршерінгу»

Виконавиця: Єфремова Дар'я Євгенівна

Керівник: к.т.н., доцент, Колісник Олена Василівна

Нормоконтролер:_____ Ігор РАЙЧЕВ

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«___»_____2021 р.

ЗАВДАННЯ

на виконання дипломної роботи

Єфремової Дар'ї Євгенівни
(прізвище, ім'я, по батькові)

1. Тема роботи: «WEB-додаток для надання послуг особистого каршерінгу» затверджена наказом ректора від «12» жовтня 2021 р. за № 2228/ст.

2. Термін виконання роботи: з 12.10.2021 по 31.12.2021.

3. Вихідні дані до роботи: теоретичні відомості та основи створення WEB–додатку, проектування та розробка WEB–додатку на основі PHP фреймворку Laravel.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, огляд та аналіз предметної області, постановка задачі, проектування та розробка WEB–додатку, висновок.

5. Перелік обов'язкового графічного матеріалу: представлений у вигляді презентації, підготовленої в PowerPoint.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання на дипломну роботу та побудова плану-графіку виконання робіт.	12.10.2021 – 15.10.2021	
2.	Аналіз існуючих методів та інструментів для створення WEB–додатку.	16.10.2021 – 19.10.2021	
3.	Проаналізувати літературу та джерела за темою дипломної роботи.	20.10.2021 – 24.10.2021	
4.	Проектування та розробка дизайну WEB–додатку.	25.10.2021 – 31.10.2021	
5.	Написання першого розділу дипломної роботи.	01.11.2021 – 13.11.2021	
6.	Обрання всіх можливих інструментів та програм для виконання задач. Написання другого розділу дипломної роботи.	14.11.2021 – 25.11.2021	
7.	Написання третього розділу дипломної роботи. Створення програмної частини роботи. Завершення створення пояснювальної записки дипломної роботи.	26.11.2021 – 10.12.2021	
8.	Оформлення та друк пояснювальної записки.	10.12.2021 – 12.12.2021	
9.	Створення презентації, доповіді та підготовка до захисту дипломної роботи.	13.12.2021 – 20.12.2021	

7. Дата видачі завдання: «12» жовтня 2021 р.

Керівник дипломної роботи_____ Олена КОЛІСНИК
(підпис керівника)

Завдання прийняв до виконання_____ Дар’я ЄФРЕМОВА
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «WEB-додаток для надання послуг особистого каршерінгу» складається із вступу, трьох розділів, висновків до кожного розділу, загальних висновків, списку використаних джерел і містить 100 сторінок тексту, 75 рисунків та 10 таблиць. Список використаних джерел містить 16 найменувань.

Метою дипломної роботи є розроблення WEB–додатку з надання послуг особистого каршерінгу для спрощення процесу оренди та зменшення кількості автомобілів у місті за рахунок спільного використання авто.

Предметом дослідження є розробка унікального WEB–додатку для надання послуг особистого каршерінгу на основі CSS, HTML, PHP, JavaScript, SQL та Laravel.

Об'єктом дослідження є організація процесу оренди та здачі власного автомобіля для збільшення прибутку.

Новизною є отримання WEB–додатку, що надає можливість здавати в оренду власне авто для більш розумного використання ресурсів.

Ключові слова: КАРШЕРІНГ, HTML, CSS, JAVASCRIPT, PHP, LARAVEL, WEB–ДОДАТОК, ФРОНТЕНД, БЕКЕНД, БАЗА ДАНИХ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Каршерінг	–	Спільне використання автомобіля
JS	–	Java Script (мова програмування)
HTML	–	HyperText Markup Language (мова тегів)
CSS	–	Cascading Style Sheets (мова стилю сторінок)
PHP	–	Hypertext Preprocessor (мова програмування)
WEB–додаток	–	Додаток, в якому клієнтом є оглядач Інтернету
WEB–сервер	–	Сервер, що приймає HTTP–запити від клієнтів
Фреймворк	–	Програмний каркас для розробки складних систем
URL	–	Uniform Resource Locator (визначає розташуванняресурсу)
CMS	–	Content Management System (систем управління контентом)
API	–	Application Programming Interface (набір методів для взаємодії компонентів)
OAuth	–	Відкритий стандарт авторизації
БД	–	База даних
SVG	–	Scalable Vector Graphics (формат файлів двомірної векторної графіки)
Репозиторій	–	Місце, де зберігаються та підтримуються дані
XAMPP	–	Локальний вебсервер
IDE	–	Integrated Drive Electronics (комплекс програмних засобів для розробки програмного забезпечення)
CRUD	–	Чотири базові функції, що використовуються під час роботи з базами даних
Фронтенд	–	Створення структури гіпертекстового документа
Бекенд	–	Програмно-апаратна частина сервісу
PDO	–	Сучасний модуль для роботи з базою даних

ВСТУП

Актуальність теми. Останнім часом людство почало все відповідальніше ставитись до використання ресурсів Землі. Нове покоління все більше переконано в тому, що немає сенсу купувати щось, що більшість свого часу буде лежати без діла, адже нам не потрібен дриль, нам потрібен кінцевий результат – отвір в стіні. Те ж саме з автомобілем: нам не потрібно його купувати, а потрібно з комфортом дібратись з однієї точки в іншу.

Сьогодні молодь не прагне купити авто, також через те, що розуміє, що це не вигідно з економічної точки зору. Адже коли ти володієш майном, майно теж володіє тобою (його потрібно обслуговувати). Володіння зараз називають словом «owening» – той що зобов'язує до певних труднощів, а от «sharing» – альтернативна модель володіння та користування річчю, так званий, обмін. Каршерінг (обмін авто) – дозволяє володіти автомобілем разом з іншою людиною, тобто володіння часткою авто з правом користуватися [1].

Так, дійсно, власна машина – це вигідно при частому подоланні великих відстаней, але з іншого боку вона 90% часу займає паркувальне місце, що і змушує задуматись володарів авто над більш раціональним використання цього ресурсу. Віддавати власний автомобіль в оренду, так само просто як і здавати власну квартиру. Тоді чому б не зробити так, щоб всі бажаючі могли економити власні кошти за рахунок зменшення витрат на одноосібне володіння авто?

Основним завданням в створенні додатку особистого каршерінгу є створення єдиного сервісу, який надаватиме всім власникам авто можливість пошуку орендарів з прийнятними навиками водіння, а орендарям – пошук авто за відповідними критеріями, а також реєстрація профілю в системі, встановленні цін за послуги, віддалене керування, отримування відгуків тощо.

Для найбільш зручного користування, даний сервіс буде представлений у вигляді WEB–додатку. Основна відмінність між WEB–додатком та додатком в тому, що перший не потрібно скачувати, адже клієнт може взаємодіяти з

WEB–сервером за допомогою браузера, а відмінність між WEB–додатком та WEB–сайтом – це можливість інтерактивної взаємодії клієнта та серверу, адже сайт – це лише джерело інформації, в той час як бажаний додаток повинен дозволяти взаємодію між користувачем та сервером за допомогою прописаних сценаріїв [2].

Мета і завдання виконання дипломної роботи – докладно дослідити область розробки, ознайомитися з основними вимогами проектування та реалізації даного додатку, провести порівняльну характеристику існуючих рішень, оцінити всі можливі варіанти вирішення поставлених задач та спрогнозувати можливість виходу даного додатку на світовий ринок.

Предметом дослідження є створення унікального WEB–додатку для надання послуг особистого каршерінгу на основі CSS, HTML, PHP, SQL та JavaScript.

Практичне значення отриманих результатів. Дана робота призначена для створення WEB–додатку, який однаково працюватиме на всіх браузерах та їх версіях.

Особистий внесок випускника – розробка унікального WEB–додатку для надання послуг особистого каршерінгу з використанням CSS, HTML, PHP, SQL та Javascript.

Виходячи з вище сказаного були поставлені конкретні ***цілі***, які мають бути реалізовані, а саме:

- Аналіз та створення основного функціоналу, на основі якого буде розроблено майбутній продукт.
- Проектування зрозумілого та простого інтерфейсу, який буде простим для користувачів, що не мають достатнього досвіду в користуванні подібним WEB–застосунком.
- Проектування WEB–додатку на базі новітніх технологій з використанням загальноприйнятих норм написання коду та побудови структури, з ціллю подальшого удосконалення.
- Реалізація бази даних.

Таким чином, створення WEB–додатку каршерінгу власного авто, що являється зручною системою для здачі машини друзям чи сусідам, є актуальним, так як це допоможе прибрати з вулиць зайві автомобілі, отримати прибуток власникам авто та заощадити орендарям власні кошти.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні відомості про каршерінг

На сьогоднішній день каршерінг є одним з глобальних напрямків розвитку економіки спільного використання речей, коли населення відмовляється від придбання благ у власність, щоб не нести відповідальність і витрати, але продовжує мати доступ до всіх досягнень наукового прогресу, використовуючи їх спільне споживання. Спільне використання авто є одним із методів ефективного використання та зберігання енергії, отриманої з альтернативних джерел.

Каршерінг уперше з'явився у Цюріху у Швейцарії в середині минулого століття. Один із місцевих кооперативів із автопарку компанії став давати своїм співробітникам в оренду машини за невелику плату і на короткий час. Через двадцять років було зроблено спробу запустити цю послугу на комерційній основі. Однак ідея не була досягнута через слабку технічну готовність її реалізувати – для клієнтів не було передбачено вільного доступу до машин, а для орендодавців була відсутня можливість контролювати їх використання [3].

Поява електронних систем навігації стала відправною точкою нової сторінки в історії розвитку каршерінгу. Завдяки можливості власникам автомобілів відстежувати їх пересування, а орендарям здійснювати до них швидкий доступ, послуга щохвилинної оренди машин стала активно поширюватися. Саме тому система короткочасної оренди машин у наші дні широко представлена в усьому світі. Число користувачів цією послугою дося-

гає близько 3 мільйонів осіб. Серед них ті, хто, виходячи з доходів, не може дозволити придбання автомобіля і при цьому не хоче брати на себе кредитні зобов'язання, люди, яким не потрібен щодня особистий транспорт і тому вони не хочуть вкладати значні суми в його обслуговування, а також мешканці міст із щільним дорожнім трафіком та труднощами паркування.

Каршерінгом користуються і особи, і представники невеликих компаній. Послуги щохвилинної оренди автомобілів, згідно з прогнозами дослідницької групи «Frost & Sullivan», у найближчі два роки зростуть майже вдесятеро, а кількість клієнтів сягатиме 26 мільйонів [3].

Популярність короткочасної оренди автомобілів пояснюється її вигодою для споживачів, яка зумовлена такими причинами:

- зменшення витрат власників особистих автомобілів, пов'язаних із зростанням цін на бензин та їх технічним обслуговуванням;
- скорочення кількості транспорту на дорогах, що зменшує можливість утворення пробок.

Законодавство більшості країн не встановлює ніяких обмежень для водіїв, які хочуть взяти авто, достатньо мати посвідчення водія. Але, страхові компанії можуть вводити додаткові критерії, які обмежують водіїв за віком та стажем водіння.

1.2. Призначення та область застосування предметної області

Головним призначенням WEB–додатку є створення платформи для спрощення процесу оренди, зменшення кількості автомобілів у місті, які стоять без діла на паркуванні. Наприклад, один автомобіль зможуть використовувати сім'ї, що живуть в одному дворі, або ж колеги чи друзі, які працюють в одному офісі.

Відповідно до статистики мешканці міста користуються автомобілем близько 3-4% від усього часу володіння. Також автомобіль займає набагато більше простору на міських дорогах, ніж громадський транспорт. І якщо вже

автомобіль займає так багато місця, то й використовуватися він повинен постійно, а не 3% на день. Весь час, що автомобіль простоює, він використовується неефективно. Додаткове споживання машин за рахунок шерінгової економіки – це, безумовно, плюс, адже таким чином, автомобілів у місті поменшає, дороги розвантажаться, що позитивно позначиться і на екологічній ситуації в місті.

Також хотілося б звернути увагу на відмінність прокату та шерінгу авто. Для прокату необхідно в робочий час відвідати офіс для обрання відповідної машини та узгодження домовленості підписанням документів. З однієї сторони каршерінг не має такого рівня сервісу як прокат, що звісно є мінусом, так як авто може бути в неналежному стані після попереднього користувача. Проте, шерінг має більшу перевагу над прокатом в тому плані, що автівки розміщені по всьому місту, саме тому їх можна орендувати чи то вдень, чи то вночі, не відвідуючи офіс, адже авто може бути припарковано поруч з Вашим місцезнаходженням. Відмикання дверей керується за допомогою додатку. Повернути авто також можна швидко та зручно, залишивши авто в буд-якому місці з ключами всередині. Варто також зазначити, що шерінг є набагато економнішим варіантом ніж прокат.

Звичайно, багато власників морально не готові довірити комусь свій автомобіль. Але є велика кількість людей, які хочуть користуватися новими технологіями та відкривають нові можливості.

1.3. Аналіз переваг і недоліків найпопулярніших реалізацій

Каршерінг доволі розповсюджена сфера в світі, тому існує багато схожих додатків, але слід зауважити, що конкуренція на початкових етапах виходу на ринок є невеликою, так як в усіх них відсутній головний інноваційний функціонал, а саме: здача власної машини в оренду. Але, як відомо з попереднього досвіду, найбільш вдалою стратегією планування нового продукту є отримання інформації з інших схожих додатків. Тому при

його розробці необхідно дослідити подібні застосунки, які були реалізовані раніше, та дослідити їх на предмет недоліків та основних переваг. Отриману інформацію можна буде використати для створення більш якісного продукту та можна позбутися тих недоліків, які є у реалізаціях інших додатках.

Переважна більшість існуючих додатків каршерінгу передбачає обов'язкову реєстрацію користувачів в системі, пошук машини за певними критеріями, резервацію обраної автівки на певний проміжок часу, а також керування процесом за допомогою мобільного пристрою. Основною ж функцією всіх існуючих додатків шерінгу є обирання бажаної автівки на основі її технічних характеристик та ціни.

На сьогоднішній день великі компанії заполонили ринок каршерінгу, через що виникає багато бюрократичних нюансів. Також суттєвим мінусом є те, що користувачі стикаються з приблизно однаково високими цінами на послуги, тим самим, не маючи великого вибору, під час підбору для себе найбільш оптимального варіанту.

Перейдемо до розгляду існуючих реалізацій додатку, що містять в собі функціонал пошуку автомобіля та орендування його на певний час.

1.3.1. Getmancar

Getmancar – єдиний існуючий сервіс каршерінгу в Україні, що являє собою мобільний додаток пошуку автомобілів для оренди. В додатку наявний широкий спектр послуг: реєстрація в системі, пошук авто різних класів, оренда на будь-який термін та пошук авто за розташуванням. Також користувач може залишити відгук, за потреби. Сервіс представлений у вигляді Android та iOS додатків [4]. На рис. 1.1 зображено приклади використання системи у мобільному додатку.

Переваги:

1. Приємний та зрозумілий інтерфейс.
2. Широкий вибір автівок за різними критеріями пошуку.
3. Швидкий пошук авто.
4. Низькі ціни.
5. Цілодобова робота сервісу.
6. Зручний пошук авто на карті.
7. Можливість залишати відгуки та оцінювати якість послуг.
8. Оновлення актуальної інформації щодо доступності авто на карті.
9. Пальне та мийка за рахунок компанії.

Недоліки:

1. Оренда авто доступна лише в Києві та Дніпрі.
2. Необхідність скачувати додаток, так як сервіс не доступний на ПК.
3. Доступність авто не в усіх районах міст.
4. Відсутність хорошої технічної підтримки.
5. Не найкращий стан авто.
6. Не завжди коректна робота додатку.
7. Відсутність здачі власного авто в оренду.

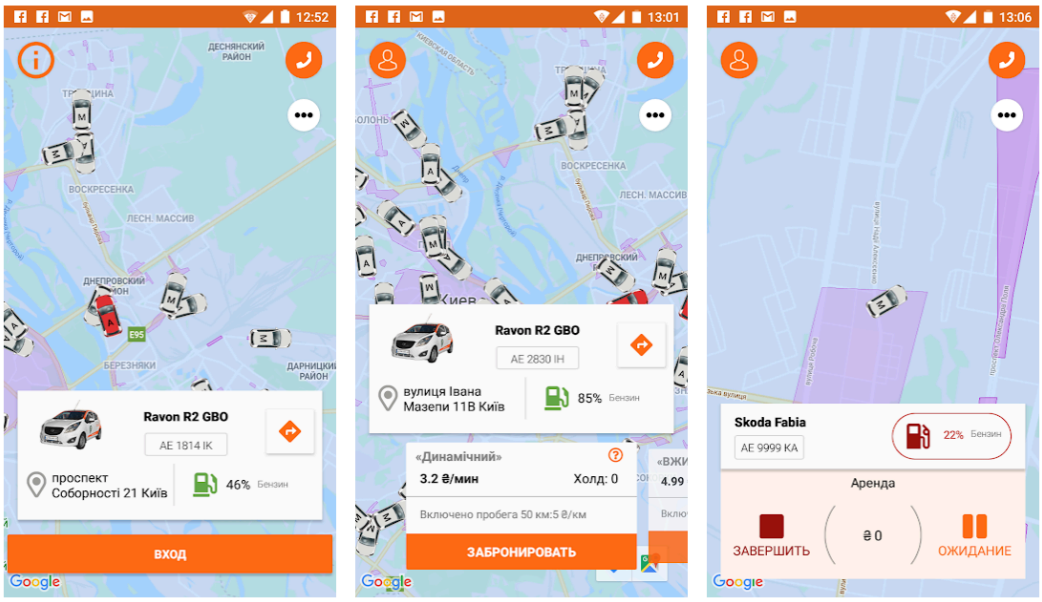

Рис. 1.1. Приклад пошуку автомобіля на мапі

1.3.2. DriveNow

DriveNow – німецький сервіс каршерінгу, власником якого є фірма BMW. DriveNow керує понад 6000 транспортними засобами в дев'яти європейських країнах. Окрім п'яти міст Німеччини (Берліна, Гамбургу, Мюнхена, Дюссельдорфа та Кельна), сервіс доступний у Відні, Копенгагені, Брюсселі, Мілані, Гельсінкі та Лісабоні. Автопарк DriveNow складається виключно з автомобільних марок вироблених компанією BMW. Ціни на оренду та транспортні засоби викладені в зручній для читання сітці, що дозволяє миттєво порівнювати найкращі пропозиції, щоденні ціни та переглядати характеристики автомобіля перед бронюванням. Здійснити бронювання швидко та легко завдяки миттєвому підтвердженню [5]. Сервіс представлений у вигляді мобільного додатку на платформах Android та iOS. Приклад використання сервісу зображено на рис. 1.2.

Переваги:

1. Приємний та зрозумілий інтерфейс.
2. Зручний пошук авто на карті.
3. Ціна оренди включає страховку, пальне та парковку.
4. Швидкий пошук авто.
5. Низькі ціни.
6. Цілодобова робота сервісу.
7. За реєстрацію надається 15 безкоштовних хвилин.
8. Оновлення актуальної інформації щодо доступності авто на карті.

Недоліки:

1. Оренда авто не доступна в Україні.
2. Необхідність скачувати додаток, так як сервіс не доступний на ПК.
3. Не найкращий стан авто.
4. Відсутність здачі власного авто в оренду.

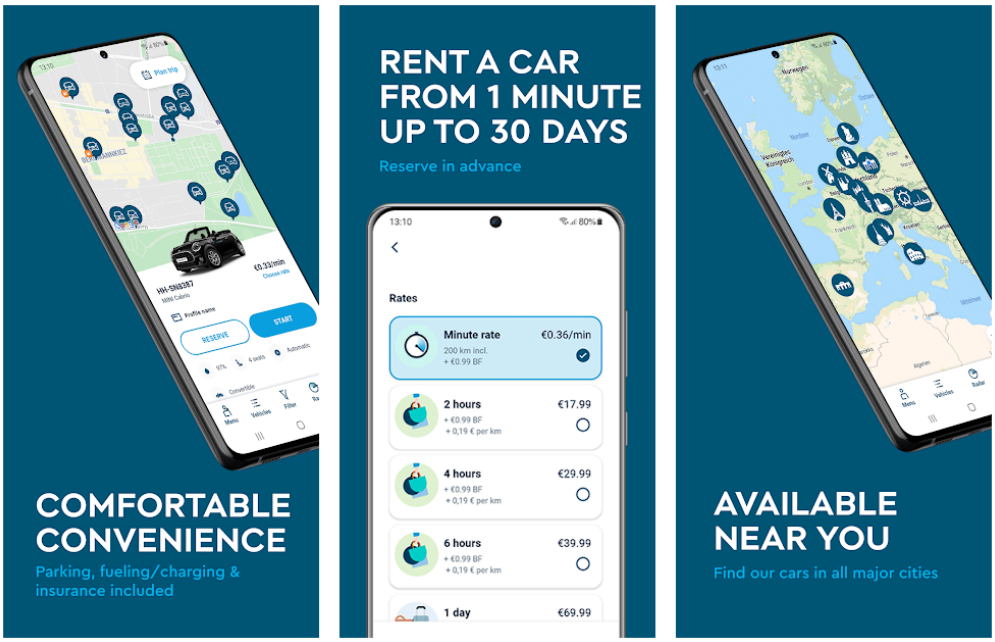

Рис.1.2. Приклад роботи додатку DriveNow

1.3.3. Herts

Herts – найбільша в світі американська компанія з прокату автомобілів, що має близько 10,2 тис. пунктів прокату у 150 країнах, у тому числі в Україні. Додаток дозволяє швидко орендувати автомобіль на день, тиждень чи місяць, переглядати або змінювати бронювання, знаходити місце розташування авто, переглядати поточні деталі бронювання або змінювати майбутні бронювання, знаходити і переглядати квитанції про оренду, а також використовувати або перевіряти свої бали. Персоналізований досвід відображає ім'я, номер учасника, статус і баланс балів в одному вікні. Також користувач може легко змінити профіль, налаштування та платежі, завдяки зручному інтерфейсу. Сервіс представлений у вигляді мобільного додатку на платформах Android та iOS. Приклад використання сервісу зображено на рис. 1.3.

Переваги:

1. Найбільша доступна система каршерінгу у всьому світі.
2. Легка навігація додатком за допомогою інтуїтивно зрозумілих функцій.
3. Швидкий пошук авто.

4. Цілодобова робота сервісу.
5. Є можливість сплачувати за бронювання балами.
6. Вхід в додаток за допомогою Touch та Face ID.
7. Зручний пошук авто на карті.
8. Оновлення актуальної інформації щодо доступності авто.

Недоліки:

1. Необхідність скачувати додаток, так як сервіс не доступний на ПК.
2. Відсутність здачі власного авто в оренду.
3. Інтерфейс лише англійською мовою.
4. Високі ціни в порівнянні з іншими сервісами.

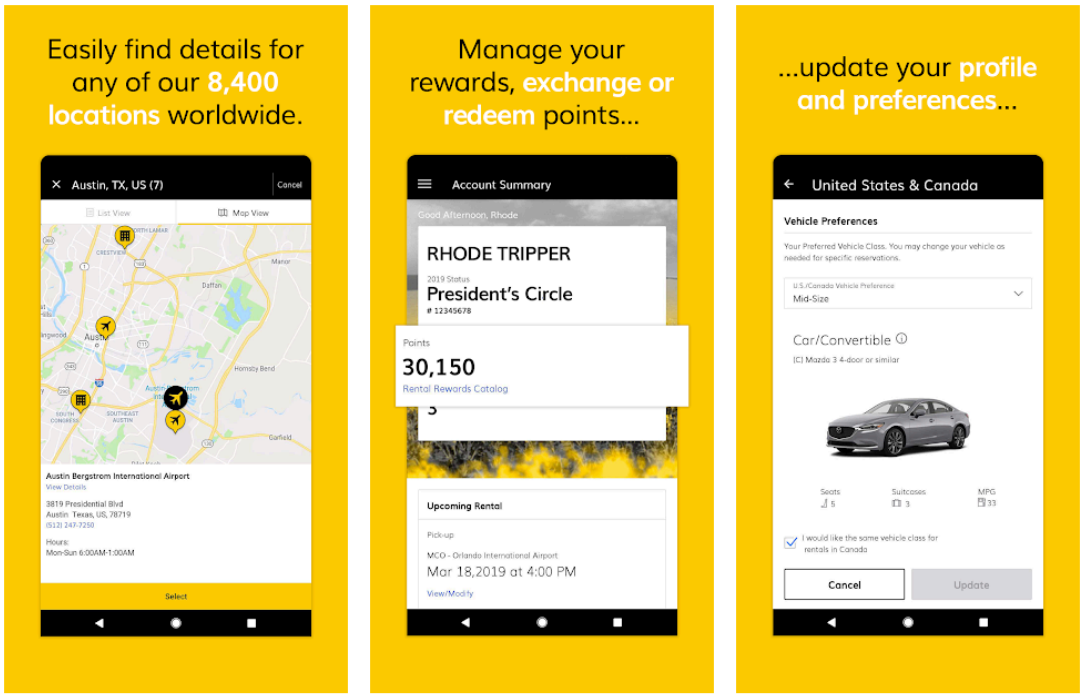

Рис.1.3. Приклад роботи додатку Herts

1.4. Порівняльна характеристика

Для вирішення проблем та недоліків існуючих систем, необхідно відокремити спільні речі, які зустрічаються найбільше та надати їм відповідний пріоритет актуальності вирішення. Порівняння буде відбуватися наступним чином: за описаними вище перевагами та недоліками існуючих додатків буде побудована відповідна таблиця, де в першій колонці займуть

місця назви додатків, а наступними колонками будуть такі характеристики, як україномовний інтерфейс, наявність функціоналу здачі власного авто, легка доступність додатку, зручний пошук за критеріями, різноманітність пошуку, зручність орендування, масштабованість на ринку, якість зворотного зв'язку. Кожному сервісу за відповідною характеристикою буде встановлений відповідний бал від 0 до 3. Бали будуть ставитися наступним чином:

1) **«0»** –якщо характеристика повністю відсутня у додатку;
2) **«1»** –якщо характеристика наявна, але реалізована на мінімальному рівні;
3) **«2»** –якщо характеристика реалізована добре, але створює додаткові труднощі користувачу;
4) **«3»** –якщо характеристика реалізована максимально просто, що цілком задовольняє користувача;

Останньою колонкою буде виступати сума балів для кожного додатку, а останній рядок – сума балів для кожної характеристики відповідно. На основі суми балів для характеристик буде встановлений відповідний пріоритет за такою відповідністю: чим нижча сума балів, тим вищий пріоритет, і навпаки. Порівняльна характеристика сервісів зображена у табл. 1.1.

Таблиця 1.1

Порівняльна характеристика сервісів

	україномовний інтерфейс	наявність функціоналу здачі власного авто	легка доступність додатку	зручний пошук за критеріями	різноманітність пошуку	зручність орендування	масштабованість на ринку	якість зворотного зв' язку	**Загалом**
Getmancar	3	0	2	3	2	3	1	2	**16**
DriveNow	0	0	2	3	1	3	2	3	**14**
Herts	0	0	2	3	3	3	3	3	**17**
Всього	3	0	6	9	6	9	6	6	**47**
Пріоритет	**2**	**1**	**3**	**4**	**3**	**4**	**3**	**3**	

З отриманої таблиці можна зробити наступні висновки. Найбільше балів отримав додаток Herts, і хоч в додатку відсутній україномовний інтерфейс та функціонал здачі в оренду власного авто, проте майже всі інші характеристики мають оцінку **«3»**. Додаток Getmancar займає друге місце, так як має лише одну оцінку **«0»**. DriveNow зовсім не відповідає 2 характеристикам та має не різноманітний автопарк, тому посідає третє місце. Найкраще всі додатки відповідають зручному пошуку за критеріями та зручності орендування, а найгірше – наявності функціоналу здачі власного авто. Тому досліджувані характеристики отримали наступні пріоритети: *«наявність функціоналу здачі власного авто»* – 1, *«україномовний інтерфейс»* – 2, *«легка доступність додатку», «різноманітність пошуку», «масштабованість на ринку» та «якість зворотного зв'язку»* – 3, *«зручний пошук за критеріями»* та *«зручність орендування»* – 4. При розробці системи варто звернути увагу на ці параметри у порядку зростання пріоритету для максимального вирішення проблем існуючих додатків каршерінгу.

1.5. Постановка задачі

Основним завданням в створенні додатку особистого каршерінгу є створення єдиного сервісу, який надаватиме всім власникам авто можливість пошуку орендарів з прийнятними навиками водіння, а орендарям – пошук авто за відповідними критеріями.

На основі вище згаданих переваг та недоліків, розглянемо головні вимоги до розроблюваного WEB–додатку:

1. Сервіс повинен бути представлений у вигляді WEB–додатку, серверна частина, якого буде виступати центральною системою керування процесом орендування.
2. У системі можуть реєструватися за телефоном абсолютно всі користувачі – як власники автомобілів, так і бажаючі скористатися послугами каршерінгу.

3. Автовласник повинен мати змогу додати своє авто, зазначивши всю необхідну інформацію, для подальшого пошуку, таку як: марка, номерні знаки та VIN-код, клас авто, тип пального, тип трансмісії, ціну за послугу тощо. Також повинна бути доступна функція змін розташування авто, статусу «вільно» та проміжку часу на який можна буде взяти авто.
4. Власники авто повинні мати змогу відстежувати пересування авто та отримувати інформацію про повернення автівки.
5. Автовласники, що додали в профіль своє авто, можуть також брати в оренду інше авто.
6. Додаток має містити пошук автівок за такими критеріями: місце розташування, клас авто, вартість послуги, тип пильного, тип трансмісії, маркер «вільно» тощо.
7. Транспортні засоби та ціни на оренду повинні бути викладені в зручній для читання сітці, що дозволить миттєво порівнювати найкращі пропозиції, щоденні ціни та переглядати характеристики автомобіля перед бронюванням.
8. Після вибору необхідної автівки система повинна прокласти найоптимальніший маршрут, враховуючи всі фактори, які можуть вплинути на час поїздки, такі як: погодні умови, затори на дорогах та якість доріг.
9. WEB–додаток повинен давати орендарям та орендодавцям змогу переглядати сторінки профілю одне одного, аналізувати та залишати відгуки.

1.6. Висновки до першого розділу

Перший розділ містить теоретичний виклад важливих аспектів та проблеми, що досліджується, та критичний огляд наукової літератури і періодичних джерел з визначеної тематики.

В зв’язку з постійною потребою клієнта в оренді автомобіля та високою ціною сервісних центрів прокату, актуальною є задача по розробці каршерінг проекту для здачі в оренду власного авто. Власний каршерінг – дозволяє володіти автомобілем разом з іншою людиною, що є вигідною позицією, так як спільне використання авто є одним із методів ефективного використання та зберігання енергії, отриманої з альтернативних джерел. Завдяки можливості автовласникам відстежувати пересування авто, а орендарям здійснювати до них швидкий доступ, послуга щохвилинної оренди машин стала активно поширюватися.

Каршерінг призначений для:

- Людей, які виходячи з доходів, не можуть дозволити придбання автомобіля.
- Людей, які не хочуть брати на себе кредитні зобов'язання на купівлю авто.
- Людей, яким не потрібен щодня особистий транспорт.
- Людей, які не хочуть вкладати значні суми в обслуговування авто.
- Мешканців міст із щільним дорожнім трафіком та труднощами паркування.
- Людей, що хвилюються про екологічну ситуацію.

Головним призначенням WEB–додатку є створення платформи для спрощення процесу оренди, зменшення кількість автомобілів у місті, які стоять на паркуванні 96% від усього часу володіння ними. Один автомобіль зможуть використовувати сім'ї, що живуть в одному дворі, або ж колеги чи друзі, які працюють в одному офісі.

Також у першому розділі був проведений огляд існуючих додатків з каршерінгу, які є на даний момент найбільш актуальними та зручними серед користувачів України та інших країн світу. Оглянуті додатки за своєю структурою є досить схожими, але кожен з них має власну бізнес-модель. Варто зазначити, що дані додатки мають як свої переваги, так і недоліки.

Всі сервіси реалізовані у вигляді мобільних додатків, які необхідно скачувати. Основними видами функцій даних додатків є: *інформативна* – користувачі переглядають на карті або у сторінці пошуку вільні автомобілі, дізнаються їх ціни, технічні характеристики та відгуки; *навігаційна* – користувач знаходить необхідний автомобіль, а додаток пропонує найкоротший маршрут та відображає його на карті; *орендування* – вибір авто та проміжок часу, за який авто буде використовуватись, в деяких випадках оплата відбувається онлайн; *керування* – користувач за допомогою мобільного додатку контролює процесом оренди та отримує актуальну інформацію про вільні авто.

До основних недоліків вже існуючих сервісів можна віднести:

1. Залежність роботи додатку від регіону використання. Переважна більшість існуючих сервісів працюють лише у конкретних країнах, містах, регіонах. Сам інтерфейс додатків виконаний в основному однією або двома мовами. Таким чином такі сервіси не мають змогу до розширення клієнтської аудиторії по світу.
2. Неможливість автовласникам реєструвати у додатку свої автівки та отримувати з них прибуток.

Порівняльний аналіз готових рішень дозволив виявити основні переваги роботи:

1. Автомобіль не буде простоювати, а буде приносити кошти.
2. Додаток може працювати в будь-якій країні світу.
3. Процес оренди значно спроститься.
4. Скоротиться кількість транспорту на дорогах, що зменшить можливість утворення пробок та зможе покращити екологічну ситуацію в світі.

У першому розділі теж була поставлена задача. Основним завданням в створенні WEB–додатку особистого каршерінгу є програмування єдиного сервісу, який надаватиме всім власникам авто можливість пошуку орендарів з прийнятними навиками водіння, а орендарям – пошук авто за відповідними

критеріями, а також реєстрація профілю в системі, встановленні цін за послуги, отримування відгуків тощо. Для найбільш зручного користування, сервіс буде представлений у вигляді WEB–додатку, який не потрібно скачувати, що заощадить пам'ять на смартфоні.

Отже, необхідно розробити таку платформу каршерінгу власного авто, що буде відповідати вище згаданим вимогам та задовольняти в максимальній мірі потреби користувачів, що тим самим вирішить недоліки існуючих додатків та зможе з максимально легкими зусиллями вийти на конкурентний рівень на ринку. Адже, додаткове споживання машин за рахунок шерінгової економіки допоможе зменшити кількість автомобілів у місті, розвантажити дороги, що позитивно позначиться і на екологічній ситуації. Звичайно, багато власників морально не готові довірити комусь свій автомобіль, але є велика кількість людей, які хочуть користуватися новими технологіями та відкривають нові можливості.

РОЗДІЛ 2
ПРОЕКТУВАННЯ WEB–ДОДАТКУ

2.1. Визначення вимог і завдань

Опис функцій та обмежень, що накладаються на систему, називають вимогами системи, а процес формування, аналізу та перевірки цих функцій та обмежень – розробкою вимог. Розробка вимог – це процес, який включає кроки, необхідні для створення та затвердження документів, що містять інформацію про системні вимоги. Одним із етапів процесу розробки вимог є формування та аналіз вимог [6]. Загальна схема процесу формування та аналізу вимог наведена на рис. 2.1.

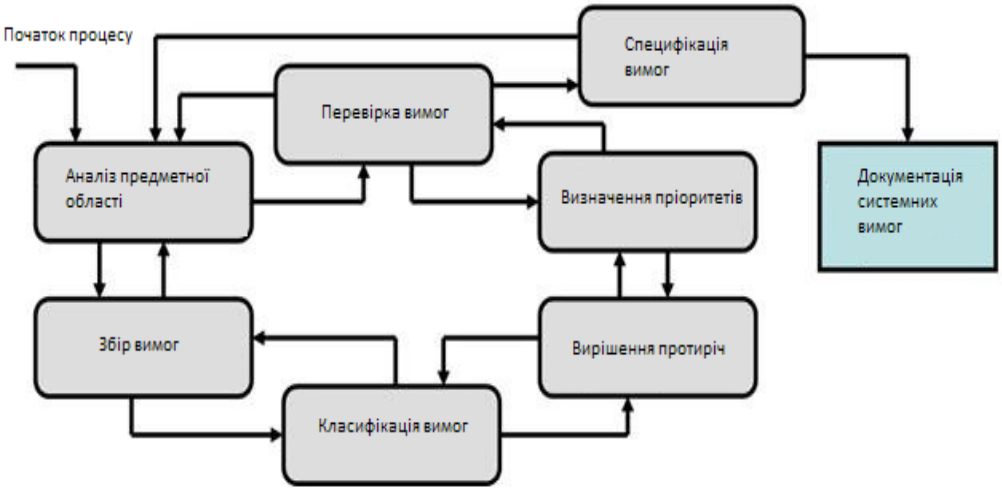

Рис. 2.1. Процес формування та аналізу вимог

Основними функціями додатку є:

1. Реєстрація в додатку профілю користувача, який хоче взяти в оренду авто з можливістю додавання інформації про власне авто.

2. Перегляд та редагування власного профілю.
3. Зміна марки «вільно» на марку «зайнято», або «вільно з..» із зазначенням періоду часу.
4. Пошук авто за наступними критеріями:
    a. марка авто;
    b. клас авто;
    c. тип пального;
    d. тип трансмісії;
    e. вартість послуги;
    f. марка «вільно»;
    g. місцезнаходження;
5. Перегляд доступних авто на мапі.
6. Прокладання маршруту до вільного авто на мапі з урахуванням наступних чинників:
    a. погодні умови;
    b. концентрація заторів на дорозі;
    c. якість дорожнього покриття.
7. Можливість залишати запит на оренду авто на обрану дату.
8. Можливість переглядати запити, що надійшли, та давати відповідь на них.
9. Можливість переходити в чат для подальшого узгодження місця та часу повернення автівки.
10. Можливість відстеження переміщення авто на мапі.
11. Можливість залишати відгуки та ставити оцінку за п’ятибальною шкалою після завершення поїздки.

Основними вимогами до додатку є:

1. Зручність та простота користувацького інтерфейсу;
2. Додаток повинен бути реалізований різними мовами.
3. Доступність системи у більшості країн світу.

2.2 Опис функціоналу системи

Працювати додаток буде наступним чином: користувачу потрібно буде зайти в WEB–застосунок та зареєструватися в ньому. Якщо у користувача є машина, то до профілю є можливість додати всю необхідну інформацію про неї. Коли автовласнику машина виявиться не потрібна, (наприклад, він вирішить поїхати у відпустку або він просто впевнений, що найближчим часом не сяде за кермо), у додатку можна буде вказати статус автомобіля – «вільний» або «вільний на певний період». Усі умови та параметри для оренди виставляє сам власник: це, наприклад, де можна забрати машину, де припаркувати після поїздки та, головне, за яку винагороду. Всі деталі бронювання заздалегідь фіксуються в онлайн-режимі. Автовласники самі зможуть обирати, кому довірити свою машину, спираючись на рейтинг орендаря в WEB–додатку, його досвід тощо.

Орендар, в свою чергу, може вибрати в пошуковій системі чи на карті оптимальний за всіма параметрами автомобіль та залишити запит. Автовласник зможе відзначити людину, якій він готовий здати свій автомобіль і в цей час орендарю надійде повідомлення, що авто є доступним, і такі автомобілісти зможуть зв’язатися між собою за телефоном чи в менеджері.

Якщо в період оренди тимчасовий власник порушить правил дорожнього руху, сплачуватиме штрафи з камер буде сам порушник, а не власник машини. Дату фіксації адміністративної провини можна буде легко порівняти з даними з оренди та визначити, хто в цей час керував автомобілем.

Функціонал додатку можна поділити на чотири основні частини: функціонал неавторизованого користувача, функціонал авторизованого користувача (орендаря), функціонал власника автомобіля та функціонал адміністратора.

Функціонал неавторизованого користувача:

1. Реєстрація профілю у системі. Для вдалої операції необхідно ввести номер телефону, на який надійде код для підтвердження

реєстрації, унікальну електронну адресу, що буде виступати логіном у додатку, своє ім’я, та прізвище, а також пароль;

2. Авторизація до системи. Для цього потрібно ввести логін і пароль, які були вказані при реєстрації профілю;
3. Перегляд основної сторінки з сіткою автівок;
4. Перегляд відображення машин на мапі;
5. Пошук необхідної автівки за різними критеріями;
6. Сортування за обраними параметрами;
7. Перегляд профілю автовласника;
8. Можливістю прокладання маршруту на мапі;

Функціонал авторизованого користувача (орендаря):

1. Перегляд та редагування профілю;
2. Перегляд профілю орендодавця;
3. Перегляд відгуків про орендодавця;
4. Перегляд основної сторінки з сіткою автівок;
5. Перегляд відображення машин на мапі;
6. Пошук необхідної автівки за різними критеріями;
7. Сортуванням за обраними параметрами;
8. Можливістю прокладання маршруту на мапі;
9. Залишення запиту на оренду авто на конкретний час;
10. Перехід в чат після підтвердження запиту;
11. Можливість залишати відгук після повернення авто;
12. Вихід з облікового запису.

Функціонал авторизованого користувача (власника автомобіля):

1. Перегляд та редагування профілю та інформації про авто;
2. Перегляд профілю орендодавця та орендаря, який надіслав запит;
3. Перегляд відгуків про орендодавця та орендаря;
4. Перегляд та підтвердження або відхилення запиту на оренду авто;
5. Зміна маркеру «вільно»;
6. Оновлення розташування власного авто;

7. Перегляд основної сторінки з сіткою автівок; на оренду авто
8. Перегляд відображення машин на мапі;
9. Пошук необхідної автівки за різними критеріями;
10. Сортуванням за обраними параметрами;
11. Можливістю прокладання маршруту на мапі;
12. Залишення запиту на оренду авто на конкретний час;
13. Можливість залишати відгук після повернення авто;
14. Вихід з облікового запису.

Функціонал адміністратора

1. Верифікація користувачів, що проходять реєстрацію у системі;
2. Блокування або видалення профілів користувачів з системи;
3. Перегляд інформації, що додається до профілю користувачів;
4. Зворотній зв’язок.

2.3. Формування вимог за допомогою діаграми прецедентів

Прецедент – це методика, яка використовується для визначення функціональних вимог системи. Робота прецеденту полягає в тому, щоб описати взаємодію між користувачами системи та самою системою, а також надати опис її функціонування. Перш ніж почати описувати прецедент, ви повинні спочатку описати сценарій [7].

Сценарій – це ряд кроків, які описують, як користувач взаємодіє з системою.

З точки зору прецеденту, користувачів називають акторами. Актор – це роль, яку користувач має можливість виконувати в системі. Один актор може виконувати кілька прецедентів і навпаки, відповідно до одного прецеденту, можуть діяти кілька акторів.

Суть діаграми прецедентів полягає в представленні розробленої системи у вигляді групи сутностей або учасників, які взаємодіють із системою за допомогою прецедентів. Учасником може бути особа, технічний пристрій,

програма або будь-яка інша система, яка може впливати на систему моделювання. Прецедентна діаграма може бути доповнена пояснювальним текстом для розкриття суті чи семантики її компонентів [7].

З урахуванням всіх вище оглянутих вимог та функцій, необхідне проведення деталізації сценаріїв користування додатком з побудовою варіативних діаграм. Акторами у даних діаграмах будуть виступати: незареєстрований користувач, власник автомобіля, орендар та адміністратор. На рис. 2.1 зображена діаграма прецедентів, яка демонструє основні взаємодії акторів та системи.

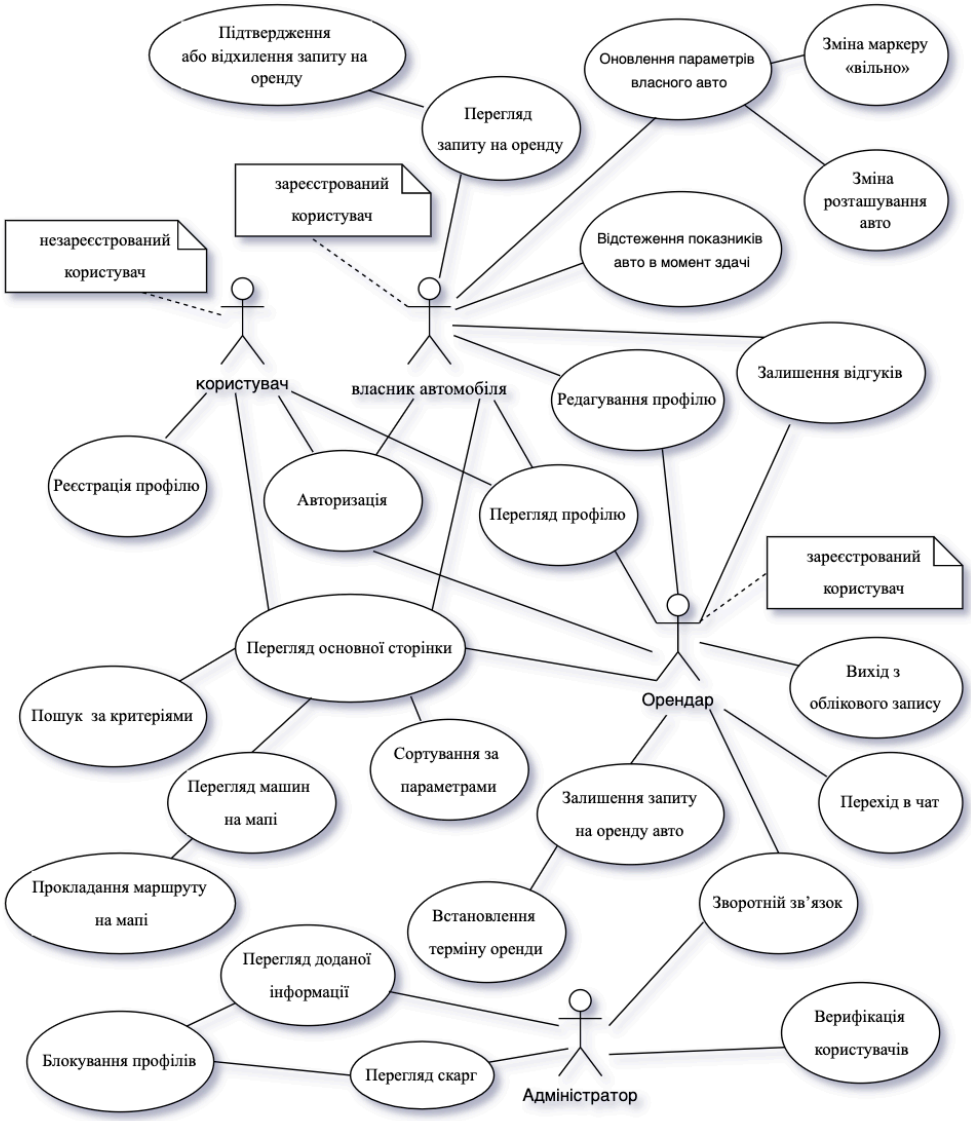

Рис. 2.2. Діаграма прецедентів взаємодії користувача і WEB–додатку

Розглянемо більш детально ієрархії окремих прецедентів для користувача – «*реєстрація*», «*авторизація*», «*редагування профілю орендодавцем*», «*редагування профілю орендарем,* «*залишення запиту на оренду авто*», «*підтвердження запиту орендодавцем*», *«процесу оренди авто*», «*залишення відгуку орендарем*», «*залишення відгуку орендодавцем*». Ієрархії зображені на рис. 2.3 – 2.11.

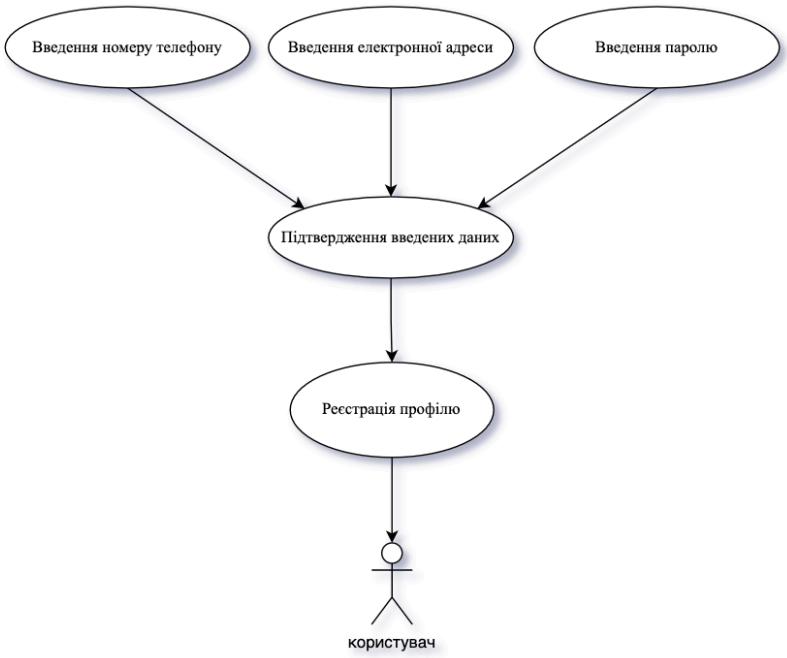

Рис. 2.3. Ієрархія прецеденту реєстрації користувача

Рис. 2.4. Ієрархія прецеденту авторизації користувача

Рис. 2.5. Ієрархія прецеденту редагування профілю орендодавцем

Рис. 2.6. Ієрархія прецеденту редагування профілю орендарем

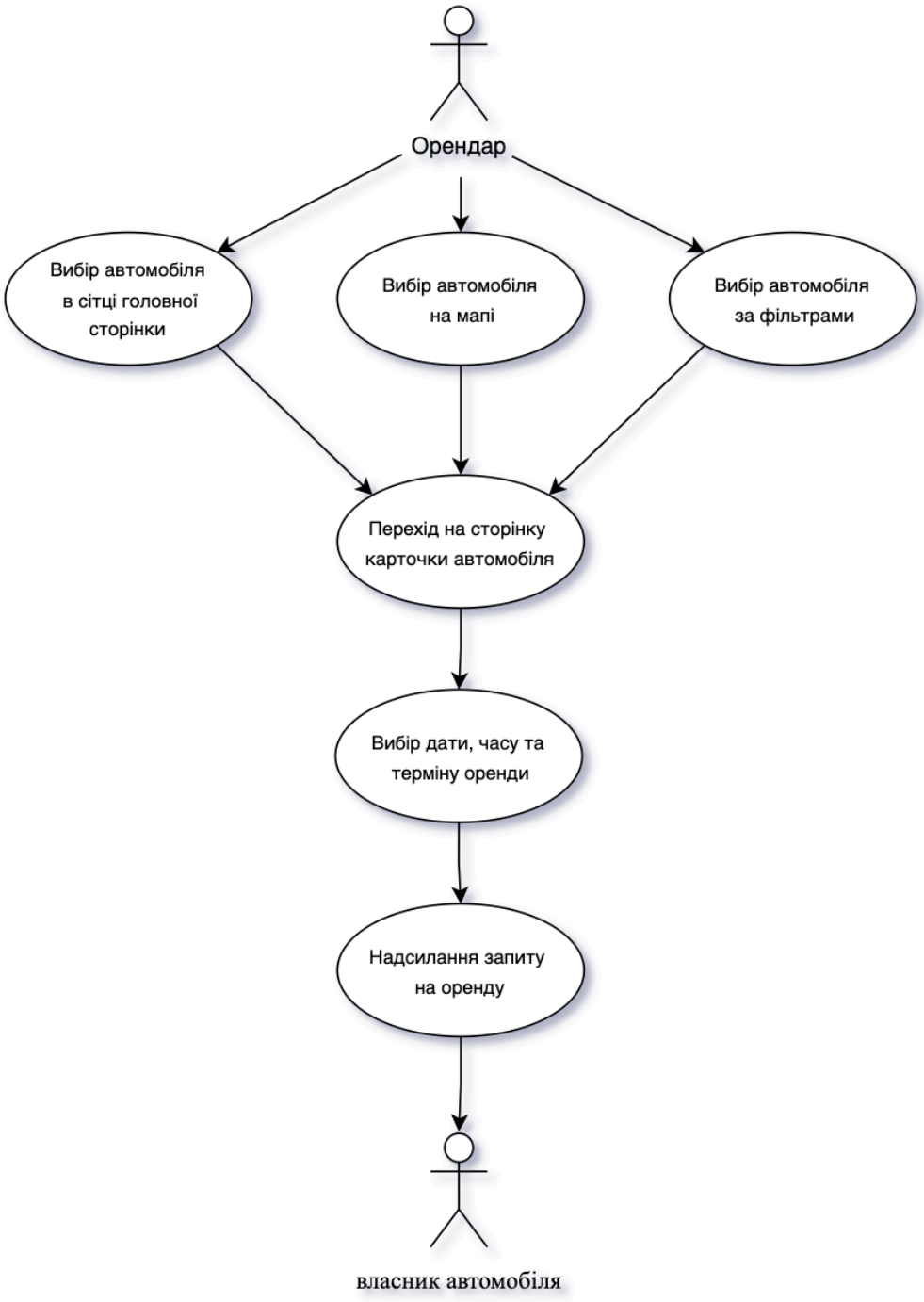

Рис. 2.7. Ієрархія прецеденту залишення запиту на оренду авто

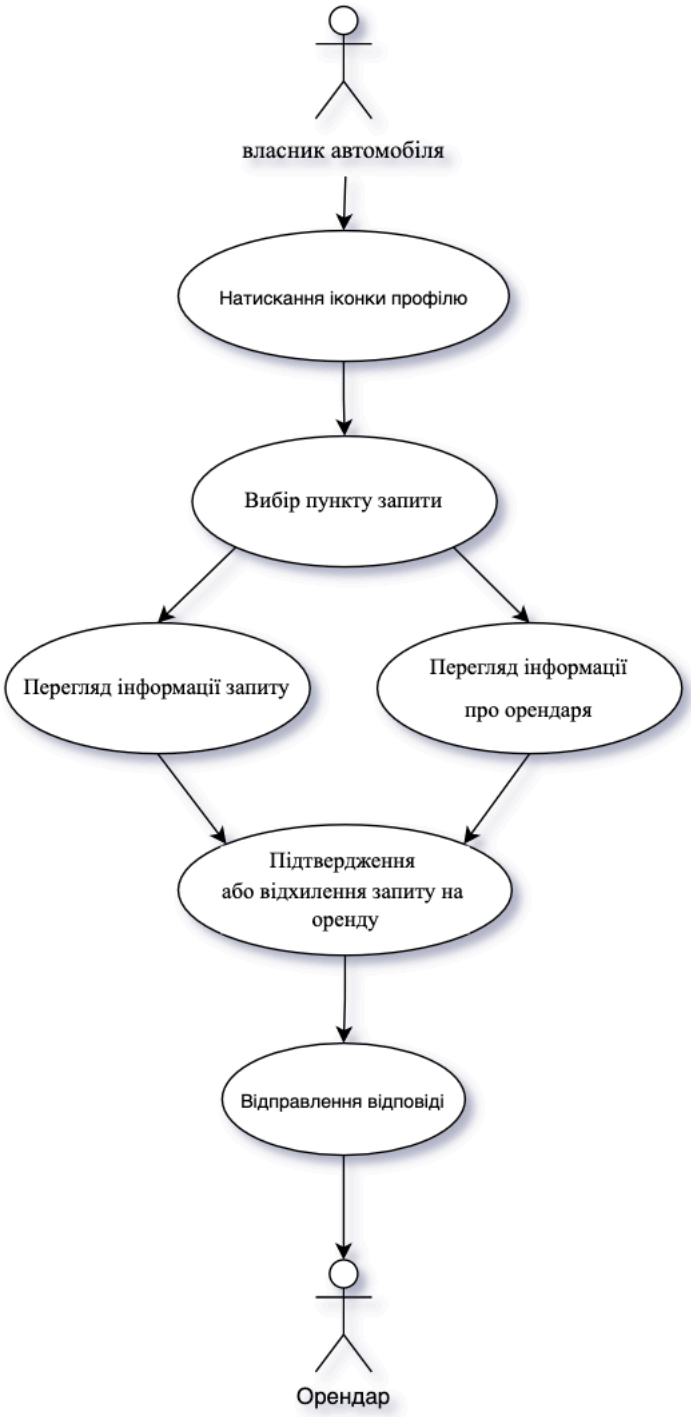

Рис. 2.8. Ієрархія прецеденту підтвердження запиту орендодавцем

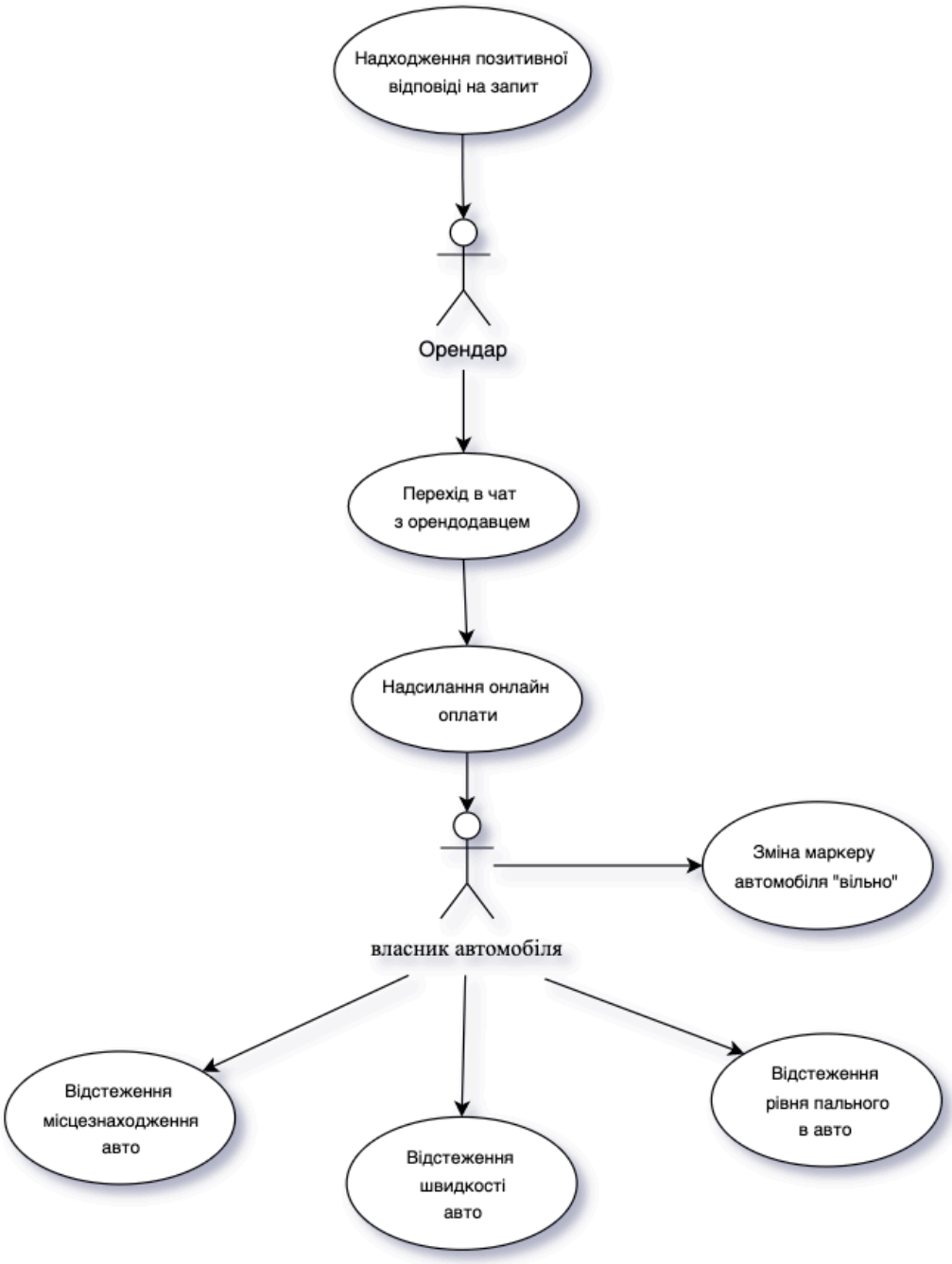

Рис. 2.9. Ієрархія прецеденту процесу оренди авто

Рис. 2.10. Ієрархія прецеденту залишення відгуку орендарем

Рис. 2.11. Ієрархія прецеденту залишення відгуку орендодавцем

Проведемо детальний огляд розвитку сценаріїв дій у розглянутих прецедентів.

2.3.1. Реєстрація

Прецедент реєстрації виступає основною дією для того, щоб користуватися основними функціями WEB–додатку. Даний прецедент можна пропустити і використовувати деякі функції сервісу як неавторизований користувач. Під час самої реєстрації необхідно ввести номер телефону, на який надійде код для підтвердження введених даних, електронну адресу, яка буде виступати логіном під час авторизації, а також вигадати пароль для входу. Електронна адреса повинна бути унікальною, в іншому випадку система згенерує повідомлення про помилку. Сценарій реєстрації представлений у табл. 2.1.

Таблиця 2.1

Реєстрація користувача

Назва	Реєстрація користувача	
Учасники	Незареєстрований користувач, система	
Передумови	Користувач неавторизований у системі	
Результат	Користувач зареєстрований	
Дії	Незареєстрований користувач	Система
1.	Введення номеру телефону, електронної адреси та паролю. Натискання на кнопку «Зареєструватись»	
2.		Перевірка правильності отриманих даних та надсилання смс коду підтвердження на номер телефону
3.	Ведення отриманого коду підтвердження. Натискання на кнопку «Зареєструватись»	

Продовження таблиці 2.1

4.		Реєстрація нового користувача в системі. Повідомлення про успішну реєстрацію
Виключні ситуації	Користувач за вказаним номером телефону або електронною адресою вже існує, занадто простий пароль	

2.3.2. Авторизація

Прецедент авторизації відбувається для того, щоб ідентифікувати користувача, надати йому доступ до певних дій в залежності від його ролей у системі. Авторизація користувача представлена у табл. 2.2.

Таблиця 2.2

Авторизація користувача

Назва	Авторизація користувача	
Учасники	Неавторизований користувач, система	
Передумови	Користувач має бути зареєстрованим у системі	
Результат	Авторизація відбулася успішно	
Дії	Неавторизований користувач	Система
1.	Введення логіну і паролю. Натискання на кнопку «Увійти»	
2.		Перевірка на коректність введених даних
3.		Повідомлення про успішний вхід. Перехід на головну сторінку
Виключні ситуації	Користувача з введеним логіном не існує, неправильно введений пароль, користувач заблокований адміністратором	

2.3.3. Редагування профілю орендодавцем

Даний прецедент актуальний в більшості випадків для тих користувачів, які бажають робити бізнес на послузі здачі в оренду власного авто. Роль орендодавця не виключає можливості орендування інших автомобілів.

Основною вимогою, щоб стати орендодавцем, є наявність всіх необхідних документів на машину. Прецедент редагування профілю орендодавцем представлений у табл. 2.3.

Таблиця 2.3

Редагування профілю орендодавцем

Назва	Редагування профілю орендодавцем	
Учасники	Авторизований користувач, система	
Передумови	Користувач має бути авторизованим у системі	
Результат	Можливість користувача здавати в оренду додане авто, відкриття нових функцій та сторінок всередині Web–додатку	
Дії	Користувач	Система
1.	Натискає іконку профілю	
2.		Відкриття нової сторінки, яка надає можливість редагування даних
3.	Введення інформації про власника (прізвище, ім’я, по-батькові, водійські права, досвід водіння), а також введення інформації про автомобіль (марка, номерні знаки, вартість за годину, тощо). Також користувач додає фотографію стану автомобіля	
4.		Обробка надісланих даних. Присвоєння користувачу ролі орендодавець, що розширює функціонал профілю. Повідомлення про результат вдалої операції. Відкриття сторінки профілю

Продовження таблиці 2.3

Виключні ситуації	Вказані не всі обов’язкові поля для заповнення, некоректні фото та введені дані, раптовий вихід з додатку

2.3.4. Редагування профілю орендарем

Даний прецедент актуальний для користувачів, які не мають власного автомобіля, а бажають орендувати наявні в системі авто.

Основною вимогою, щоб стати орендодавцем, є наявність водійських прав та досвідом водіння авто. Прецедент редагування профілю орендарем представлений у табл. 2.4.

Таблиця 2.4

Редагування профілю орендарем

Назва	Редагування профілю орендодавцем	
Учасники	Авторизований користувач, система	
Передумови	Користувач має бути авторизованим у системі	
Результат	Можливість користувача здавати в оренду додане авто, відкриття нових функцій та сторінок всередині Web–додатку	
Дії	Користувач	Система
1.	Натискає іконку профілю	
2.		Відкриття нової сторінки, яка надає можливість редагування даних
3.	Введення інформації про власника (прізвище, ім’я, по-батькові, водійські права, досвід водіння). Також користувач додає власну фотокартку та фото водійських прав	
4.		Обробка надісланих даних. Присвоєння користувачу ролі орендаря. Повідомлення про результат вдалої операції. Відкриття сторінки профілю
Виключні ситуації	Вказані не всі обов’язкові поля для заповнення, некоректні фото та введені дані, раптовий вихід з додатку	

2.3.5. Залишення запиту на оренду авто

Прецедент залишення запиту на оренду авто є досить популярним та використовуваним у додатках з каршерінгу. Ключовими вимогами під час вибору авто є різноманітність фільтрів та величин сортування. Прецедент доступний для всіх ролей користувачів, окрім неавторизованих. Прецедент представлений у табл. 2.5.

Таблиця 2.5

Залишення запиту на оренду авто

Назва	Залишення запиту на оренду авто	
Учасники	Авторизований користувач, система	
Передумови	Користувач має заповнити всі персональні дані (водійські права та досвід за кермом)	
Результат	Відправлення запиту на оренду обраного авто	
Дії	Авторизований користувач	Система
1.	Натискає на кнопку фільтр зверху на головній сторінці	
2.		Відкриття вікна з фільтрами
3.	Вибір необхідних параметрів в фільтрі та вибір діапазонів значень фільтрів. Натискання на кнопку «Знайти»	
4.		Виконання пошуку. Відображення списку доступних автомобілів згідно з критеріями пошуку
5.	Вибір в сітці на автомобіля, що сподобався. Натискання на нього	
6.		Завантаження сторінки карточки автомобіля
7.	Вибір дати, часу та терміну оренди в календарі. Натискання на кнопку «Залишити запит»	
8.		Обробка запиту. Надсилання запиту власнику автомобіля

Продовження таблиці 2.5

Виключні ситуації	Спроба орендувати автомобіль на недоступний термін, на недоступну дату або на недоступний час

2.3.6. Підтвердження запиту орендодавцем

Даний прецедент необхідний для встановлення зв’язку власника автомобіля з орендарем перед узгодженням нюансів оренди. Орендар може відхилити запит, таким чином зв’язок не буде встановлений. Сценарій прецеденту представлений у табл. 2.6.

Таблиця 2.6

Підтвердження запиту орендодавцем

Назва	Підтвердження запиту орендодавцем	
Учасники	Авторизований користувач, система	
Передумови	Запит повинен бути надісланий орендодавцю	
Результат	Відправлення відповіді на запит орендарю	
Дії	Авторизований користувач	Система
1.	А на головній сторінці натиснути на іконку профілю в правому верхньому кутку. У випадаючому меню натиснути пункт «Запити»	
2.		Відкриття сторінки запитів
3.	Перегляд інформації запиту. .Натискання кнопки «Переглянути інформацію про орендаря»	
4.		Завантаження профілю орендаря
5.	Підтвердження або відхилення запиту на оренду	
6.		Відправлення відповіді орендарю
Виключні ситуації	Відсутні	

2.3.7. Процес оренди авто

Прецедент процесу оренди авто є основним і найбільш використовуваним. В даному випадку учасниками прецеденту є 3 ключові фігури: власник автомобіля, орендар та система. Прецедент процесу оренди авто представлений у табл. 2.7.

Таблиця 2.7

Процес оренди авто

Назва	Процес оренди авто		
Учасники	Власник автомобіля, орендар, система		
Передумови	Орендодавець повинен відправити позитивну відповідь на запит		
Результат	Оренда автомобіля відбулася успішно		
Дії	Орендар	Система	Власник автомобіля
1	2	3	4
1.	У вікні про надходження позитивної відповіді на запит натискає кнопку «Перехід чат з оредодавцем»		
2.		Відкрите чату орендаря з власником автомобіля	
3.			Повідомлення про всі нюанси оренди авто
4.	Здійснення оплати		
5.			Натискання в профілі кнопки «вільно» для її змінення
6.		Зміна кнопки на «Зайнято»	

Продовження таблиці 2.7

1	2	3	4
7.			Натискання кнопки «Оновлення інформації» для відстеження місцезнаходження
8.		Оновлення та відображення інформації	
9.	Повідомлення орендодавця про повернення авто		
10.			Натискання в профілі кнопки «Зайнято» для її змінення
11.		Зміна значення кнопки «Зайнято» на «Вільно»	
Виключні ситуації	Відсутність можливості відстеження місцезнаходження авто		

2.3.8. Залишення відгуку орендарем

Прецедент залишення відгуку є важливим для зворотного зв’язку, так як для майбутньої співпраці важлива повна інформація про якість наданих послуг. Сценарій прецеденту представлений у табл. 2.8.

Таблиця 2.8

Залишення відгуку орендарем

Назва	Залишення відгуку орендарем	
Учасники	Орендар, система	
Передумови	Завершення отримання послуг	
Результат	Публікація відгуку в профілі власника автомобіля	
Дії	Орендар	Система
1.	Натискає на профіль орендодавця	

Продовження таблиці 2.8

2.		Завантаження сторінки орендодавця
3.	Натискає на поле «Додати відгук» Пише відгук. Натискає на кнопку «Підтвердити»	
4.		Обробка відгуку. Додавання відгуку до профілю автовласника. Надсилання сповіщення власнику про надходження відгуку
Виключні ситуації	Відгук містить некоректний контент	

2.3.9. Залишення відгуку орендодавцем

Прецедент залишення відгуку є важливим для зворотного зв’язку, так як для майбутньої співпраці важлива повна інформація про якість отримання послуг. Сценарій прецеденту представлений у табл. 2.9.

Таблиця 2.9

Залишення відгуку орендодавцем

Назва	Залишення відгуку орендодавцем	
Учасники	Власник авто, система	
Передумови	Завершення надання послуг	
Результат	Публікація відгуку в профілі орендаря	
Дії	Власник авто	Система
1.	Натискає на профіль орендаря	
2.		Завантаження сторінки орендаря
3.	Натискає на поле «Додати відгук» Пише відгук. Натискає на кнопку «Підтвердити»	

Продовження таблиці 2.9

4.		Обробка відгуку. Додавання відгуку до профілю орендаря Надсилання сповіщення орендарю про надходження відгуку
Виключні ситуації	Відгук містить некоректний контент	

2.4. Формування вимог за допомогою діаграми діяльності

Іншою важливою діаграмою в UML є діаграма діяльності. Вона описує динамічні аспекти системи та являє собою блок-схему, яка відображає логіку або послідовність переходу від однієї діяльності до іншої, і фокус завжди приділяється результату діяльності. Зображення діаграми діяльності показано на малюнку. 2.12.

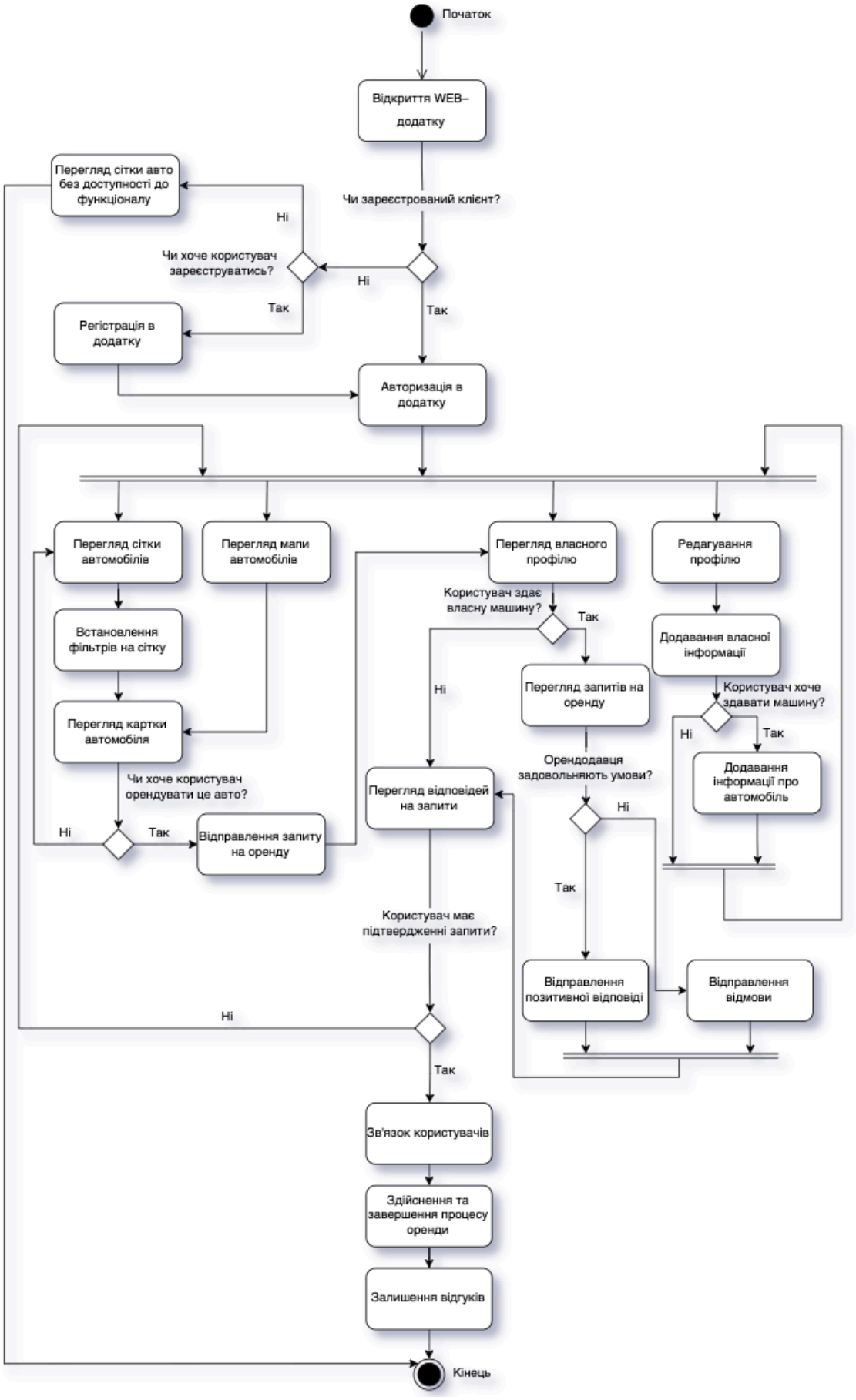

Рис. 2.12. Діаграма діяльності для розроблюваного WEB–додатку

2.5. Формування вимог за допомого діаграми послідовності

Діаграма послідовності UML – це діаграма взаємодії, яка детально описує, як виконуються операції. Вона відображає взаємодію між об'єктами в момент співпраці. Діаграма послідовності орієнтовані на час, вони візуально показують послідовність взаємодії та використовують вертикальну вісь діаграми для представлення «ліній життя» окремих об’єктів, які беруть участь у взаємодії. Діаграми послідовності використовуються для запису та розуміння вимог нових та існуючих систем. На малюнках 2.13 – 2.19 зображено діаграми послідовності для розроблюваного WEB–додатку.

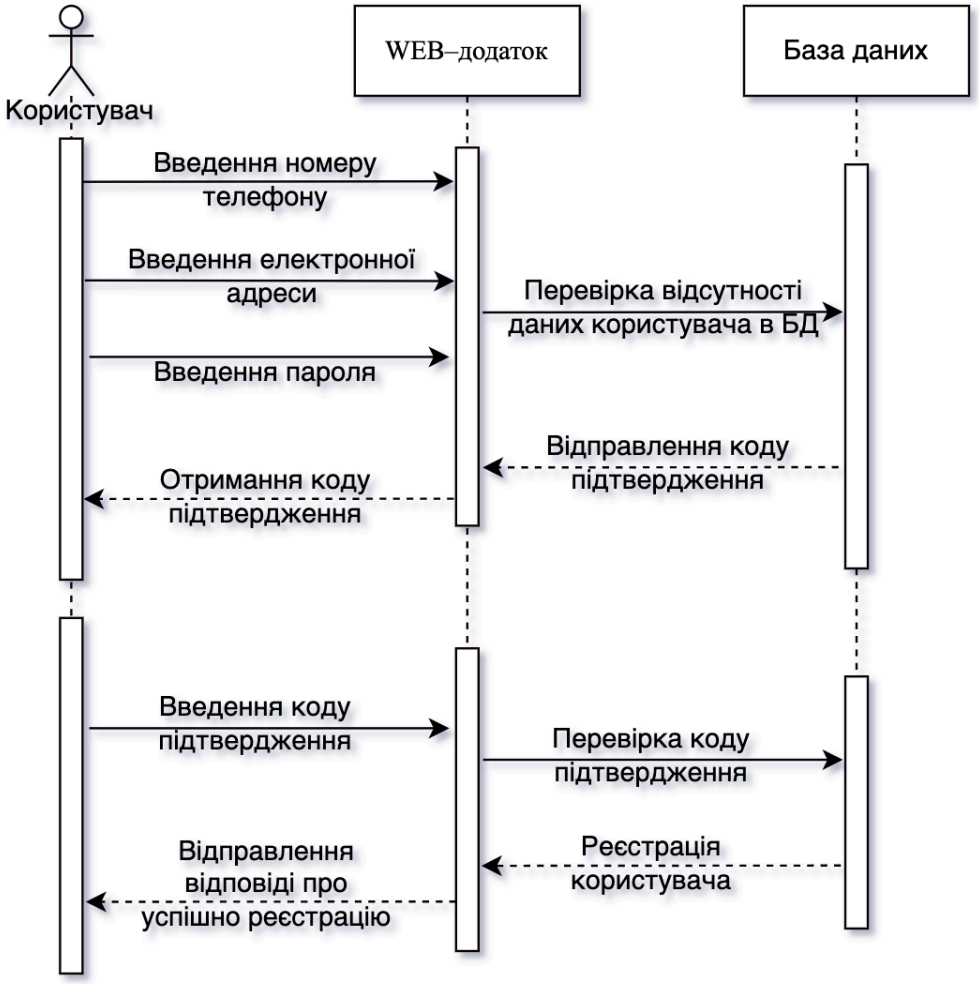

Рис. 2.13. Діаграма послідовності процесу реєстрації користувача

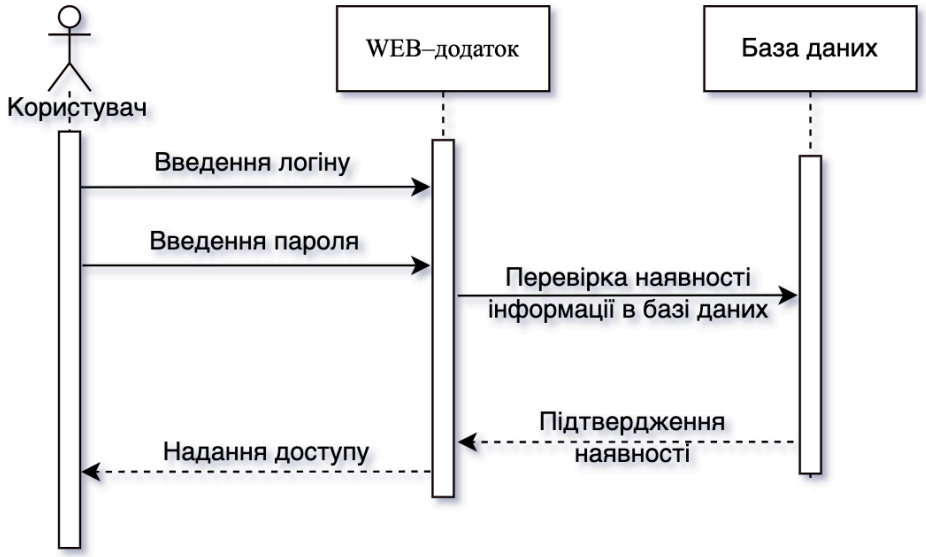

Рис. 2.14. Діаграма послідовності процесу авторизації користувача

Рис. 2.15. Діаграма послідовності процесу додавання автомобіля у WEB–додаток

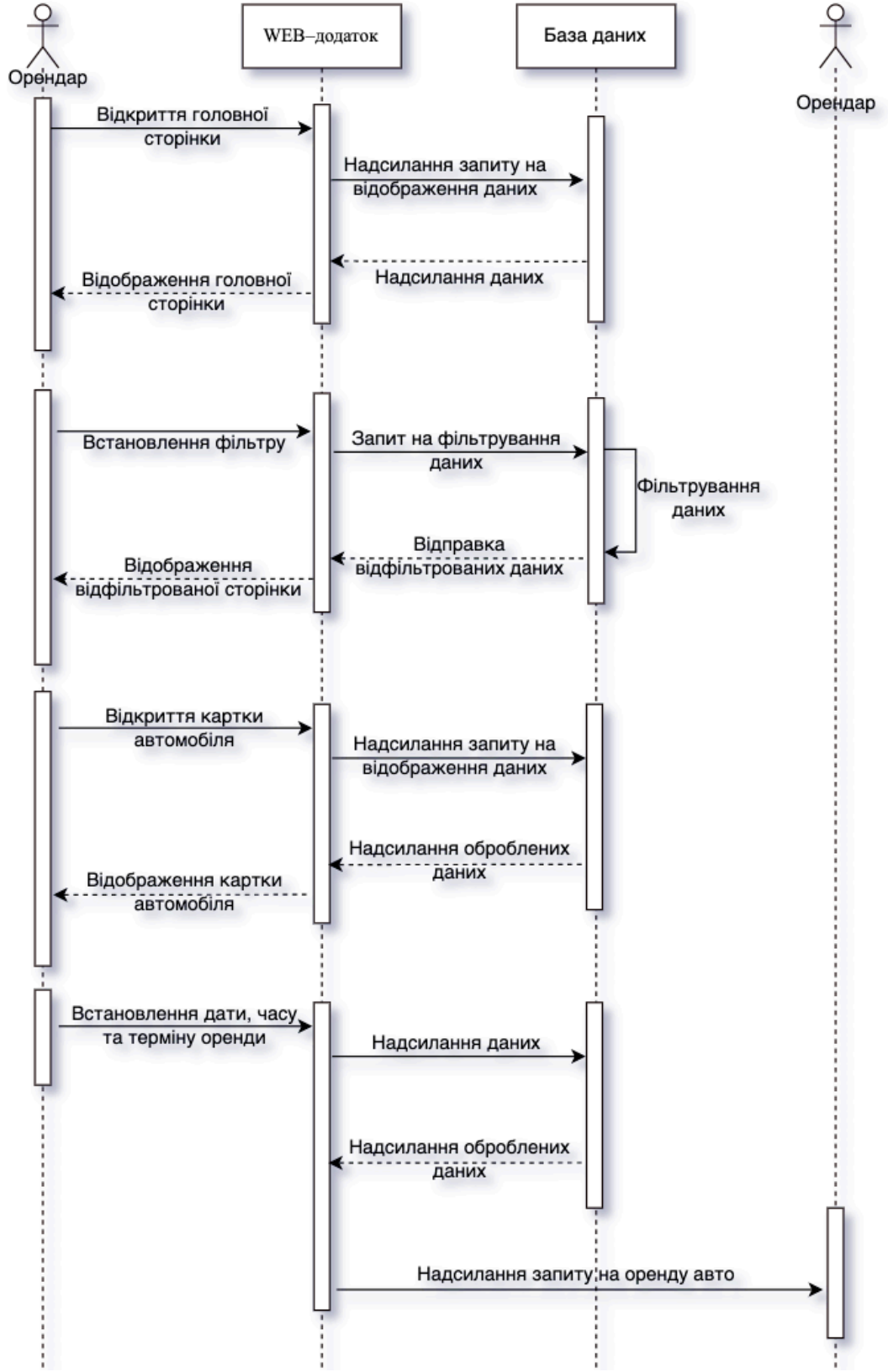

Рис. 2.16. Діаграма послідовності процесу фільтрування та відправлення запиту

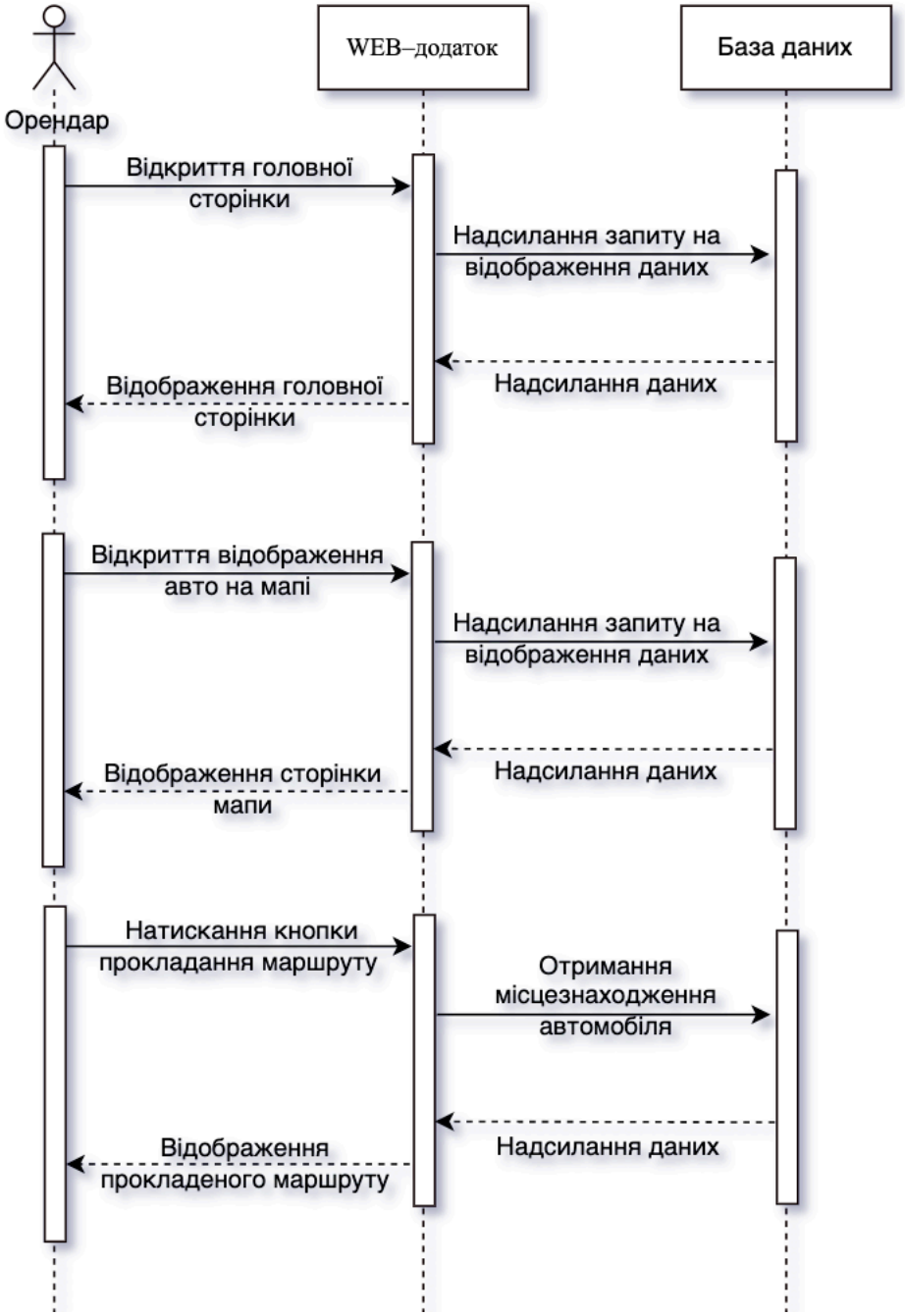

Рис. 2.17. Діаграма послідовності процесу прокладання маршруту до авто на карті

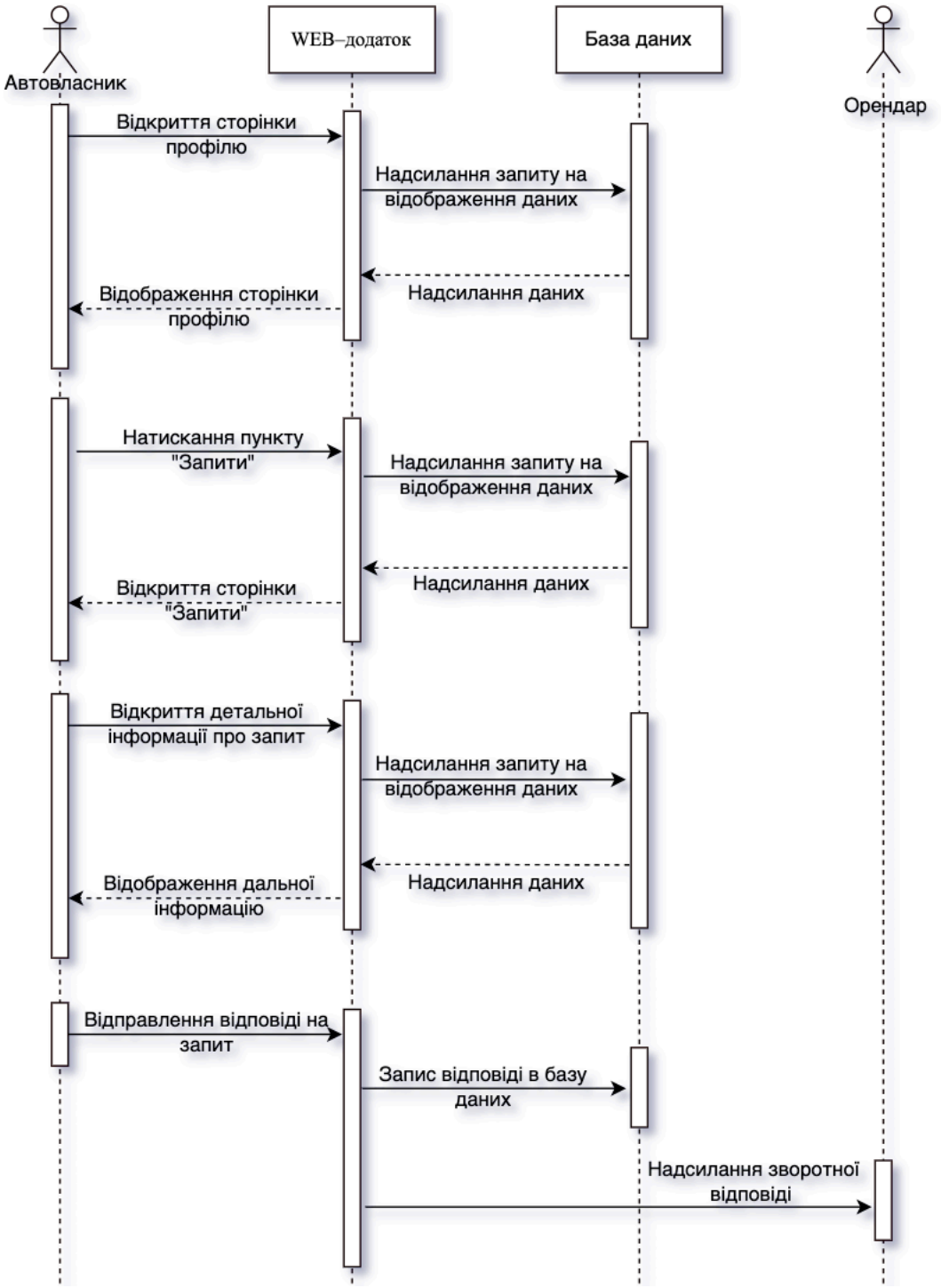

Рис. 2.18. Діаграма послідовності процесу надсилання зворотної відповіді

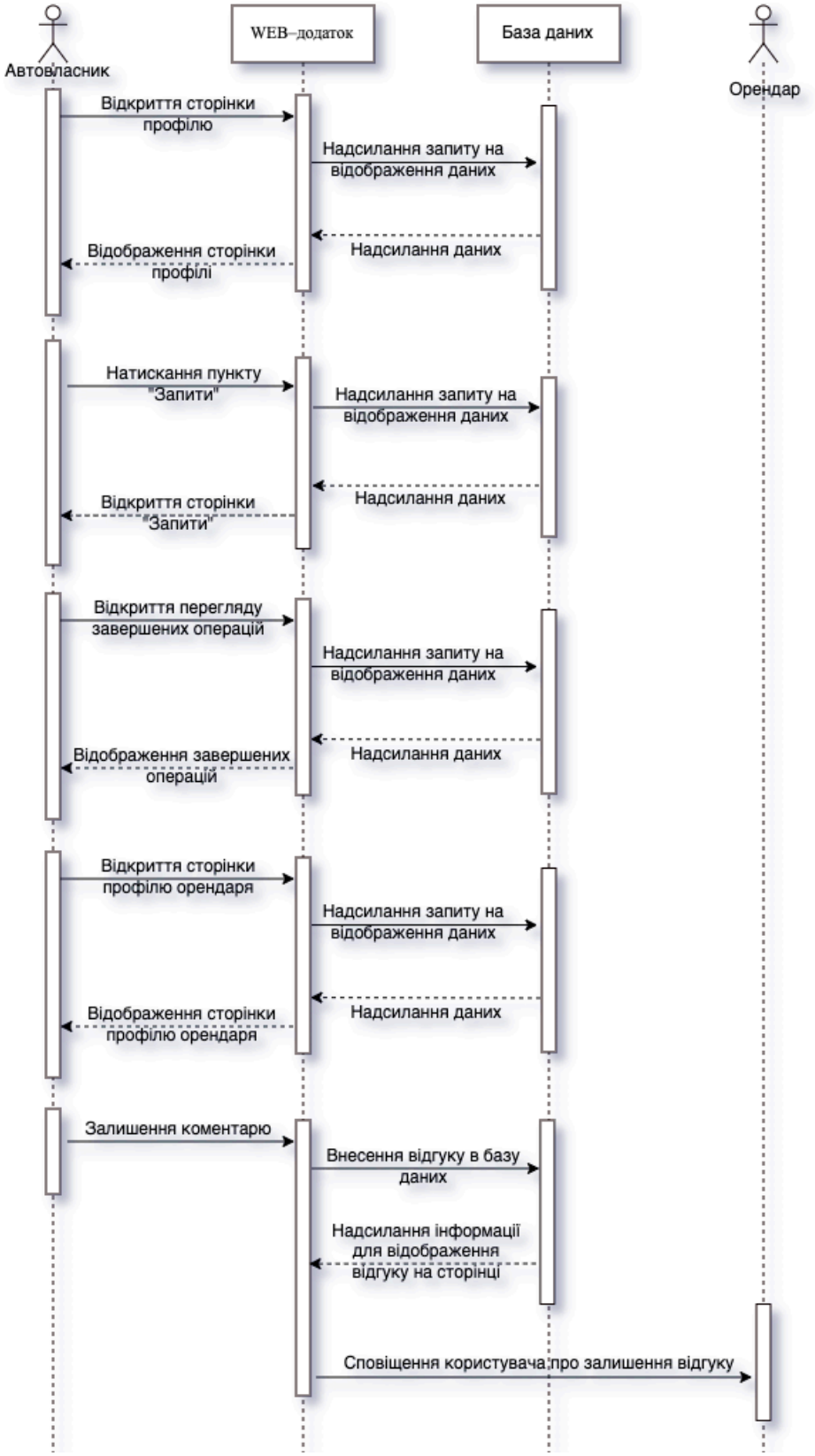

Рис. 2.19. Діаграма послідовності процесу залишення відгуку

2.6. Висновки до другого розділу

У другому розділі описується предметна область дипломної роботи та основний функціонал WEB–додатку. В процесі розгляду даного розділу було сформовано та описано основні вимоги до розроблюваного WEB–додатку з надання послуг власного каршерінгу.

Було розроблено можливі сценарії прецедентів під час виконання основних функцій додатку. Суть діаграми прецедентів полягає в тому, що розроблена система представлена у вигляді набору сутностей або акторів, які взаємодіють із системою за допомогою варіантів використання.

Також було визначено можливі виключні ситуації, які можуть негативно вплинути на функціональність роботи додатку, що дозволить користувачам комфортно використовувати додаток та уникнути можливих збоїв програми.

На основі створених сценаріїв було побудовано діаграму прецедентів для чотирьох учасників додатку: неавторизований користувач, авторизований користувач (орендар), автовласник, що хоче здавати в оренду власне авто та адміністратор. Було побудовано окремі діаграми прецедентів для більш детального розгляду основних функцій сервісу:

1. реєстрація;
2. авторизація;
3. редагування профілю орендодавцем;
4. редагування профілю орендарем;
5. залишення запиту на оренду авто;
6. підтвердження запиту орендодавцем;
7. процес оренди авто;
8. залишення відгуку орендарем;
9. залишення відгуку орендодавцем.

Ці функції були розглянуті більш детально на етапі проектування, оскільки вони взаємозалежні з іншими функціями і мають дуже великий вплив на відображення коректних даних у реальному часі.

Було побудовано діаграму діяльності, яка являє собою граф, вершинами якого є дії в WEB–додатку, а перехід відбувається по завершенню цих дій за допомогою стрілки. Заокруглений прямокутник представляє дію, ромб вказує на рішення, лінія позначає початок або кінець паралельної діяльності, чорне коло вказує на початок процесу, а чорне коло в колі вказує на кінець процесу. Стрілки показують порядок, у якому відбуваються дії від початку до кінця.

Було вирішено побудувати сім діаграм послідовності, для окремих найголовніших процесів, таких як:

1. реєстрація користувача;
2. авторизація користувача;
3. додавання автомобіля у WEB–додаток;
4. фільтрування та відправлення запиту;
5. прокладання маршруту до авто на карті;
6. надсилання зворотної відповіді;
7. залишення відгуку.

Основними елементами діаграми є позначка об'єкта (прямокутник з назвою), вертикальна «лінії життя», що відображає хід часу, прямокутники, що відображають діяльність об'єкта або деякі виконані функції та стрілка, що показує обмін повідомленнями між об’єктами. Кожен об’єкт відображається як окремий вертикальний стовпець.

Спільним для усіх розроблених діаграм є те, що кожна показує опис роботи програми. А відрізняються вони у графічному зображенні. Отже, згідно з аналізом, основними учасниками розроблюваного додатку є кінцеві користувачі додатку. Будь-який користувач, який використовує додаток, матиме безпосередній вплив на нього. Він може переглядати доступні в додатку автівки, як на мапі так і в зручній для читання сітці, також є можливість взаємодії з профілями зареєстрованих користувачів: залишати запити, відповідати на запити, залишати відгуки, а також, якщо вашу машину взяли в оренду, то доступний функціонал зручного відстеження місцезнаходження авто.

РОЗДІЛ 3
РОЗРОБКА WEB–ДОДАТКУ

3.1. Етапи створення WEB–додатку

WEB–додаток – це комп'ютерна програма, яка виконує свої функції, використовуючи WEB–браузер у якості свого клієнта. Етапи створення WEB–додатку необхідно знати для того, щоб розуміти як перебігає процес створення кінцевого продукту. Можна відокремити 6 основних етапів.

1-й етап – це виявлення потрібності. Ситуація в якій замовник звертається до розробників з певної задачею: зробити WEB–додаток. Для того щоб розробник міг виконати поставлену задачу необхідно розібратися який конкретно додаток потрібен. Наприклад, якщо замовник хоче, щоб сайт компанії був в Інтернеті – то йому потрібен сайт-візитка. Якщо він хоче продавати товари у великій кількості через Інтернет, то йому потрібен інтернет магазин. Якщо він хоче щоб його продукт або послуга продавалися з великою вірогідністю, то йому необхідний односторінковий сайт, який побудований згідно усіх маркетингових потреб, а саме: лендінг.

Вся ця інформація дізнається на етапі розмови, яка відбувається під час зустрічі з замовником, цей процес може відбуватися в мережі або наживо. Розробник задає питання, щоб визначити, що конкретно потрібно.

Також є спеціальні сервіси, які можуть створити опитування. Таке опитування найчастіше являє собою документ в якому замовник відповідає на питання розробника. В опитуванні для створення сайту заповнюється інформація про діяльність компанії, назва компанії, чи є подібні сайти, що повинно бути на сайті тощо.

Але не на всі питання замовник може дати відповідь, тому все рівно важливо підтримувати спілкування з ним, задавати питання тощо. Також можна створювати за допомогою Google forms власне опитування. Відповіді на ці питання зазвичай заповнюються в Exel табличку або будуть приходити на пошту.

Наступний етап, після того як розібрались задачею, це етап написання тексту, створення скелету майбутнього сайту, тобто бачення того, як все буде виглядати в майбутньому. На цьому етапі немає дизайну чи кольорів, все максимально просто, але має бути зрозумілий концепт. Цей етап називається етапом прототипування. Прототипування необхідно, щоб зрозуміти хто складає цільову аудиторію.

3-й етап – дизайн. Дизайнер малює концепцію в кольорах, в картинках, в розміщені елементів, в шрифтах. Це все накладається на прототип, розроблений на попередньому етапі, і вже можна побачити готову картинку. На цьому етапі зрозуміло як буде виглядати додаток та як будуть виглядати окремі елементи, щоб показати замовнику. Але це лише картинка, це не справжній сайт.

4-й етап – верстка та програмування. WEB розробник пише код, щоб оживити макет та отримати працюючий сайт, на якому натискаються всі елементи, можна додати певну інформацію або заповнити форму. Верстка реалізується за допомогою HTML, CSS та JavaScript. На цьому етапі також відбувається посадка на системи керування, наприклад WordPress. Це необхідно для того щоб замовник міг додавати та/або редагувати елементи.

5-й етап – публікація та тестування. Сайт публікуються для загального доступу (вигружається на хостинг, прив’язується домен), тестується та оптимізується швидкість його роботи. Тестування відбувається за рахунок того, що виявляються слабкі місця. Наприклад, додаток загружається занадто повільно, тому що має важкі картинки. Необхідно оптимізувати картинки, що призводить до того, що швидкість сайту збільшується. Також тестування відбувається на різних браузерах та пристроях.

Останній етап це просування та розвиток сайту. Після того як його було опубліковано і він став доступний для всіх, люди які займаються контекстною рекламою, починають просувати продукт. Ми знаємо, що в пошуковій системі перші 4 сайти – це реклама. Перші місця пошукової системи є платними. Далі йдуть сайти, які просуваються по SEO (без оплати за кожен клік).

3.2. Вибір технологій та їх обґрунтування

Проведемо огляд технологій розробки, які найкраще підійдуть для реалізації додатку.

3.2.1. Вибір платформи для додатку

Враховуючи всі вимоги, які висуваються до додатку, необхідно реалізувати систему, що буде максимально універсальною для користувача. З огляду на те, що останнім часом користувачі віддають перевагу мобільним пристроям та не хочуть заповнювати їх пам’ять, то найоптимальнішим вибором є створення кросплатформного WEB–додатку, що буде доступний як для користувачів Android, так і для користувачів iOS, так як така багатоплатформність надає можливість працювати більш ніж на одній апаратній платформі. Кросплатформність дозволяє суттєво знизити витрати на розробку нового та адаптацію існуючого функціоналу [8].

Так як це WEB–додаток, реалізація можлива за рахунок використання технологій CSS та HTML.

HTML (Hypertext Markup Language) – це код, який використовується для структурування та відображення WEB–сторінки та її контенту [9]. HTML не є мовою програмування, це мова розмітки, і використовується, щоб повідомляти браузеру, як відображати контент. Він може бути складним або простим, залежно від того, як WEB–розробник спроектує дизайн.

CSS (Cascading Style Sheets) – це код, який використовується для стилізації WEB–сторінки [9]. Як і HTML, CSS насправді не є мовою програмування. Але це не мова розмітки, а мова таблиці стилів, що використовуються до елементів у документах HTML.

3.2.2. Вибір мови програмування

Для того, щоб додати інтерактивність у WEB–додаток, таку як: відгук при натисканні кнопок, введення даних у форму, динамічні стилі або анімація, найкращим рішення буде використовувати мову програмування JavaScript. JavaScript або JS – це повноцінна динамічна мова програмування, яка застосовується до HTML документа, і може забезпечити динамічну інтерактивність у WEB–додатках. Ця мова універсальна та доброзичлива до новачків. Маючи великий досвід, за допомогою цієї мови можна створювати ігри, анімовану 2D та 3D графіку, повномасштабні програми з базами даних та багато іншого [10].

JavaScript компактна мова, але дуже гнучка. Розробниками написано велику кількість інструментів поверх основної мови, які розблоковують величезну кількість додаткових функцій. Таких як:

- Програмні інтерфейси програми (API), вбудовані в браузери, що забезпечують різні функціональні можливості.
- Сторонні API дозволяють розробникам впроваджувати функціональність у свої додатки від інших розробників.
- Можливість застосувати до HTML сторонні фреймворки та бібліотеки, що дозволить прискорити створення сайтів та програм [10].

Щоб розробити хороший WEB–додаток – треба також знати такі мови як PHP та SQL.

PHP (Hypertext Preprocessor) – це поширена мова програмування загального призначення з відкритим вихідним кодом. PHP спеціально сконструйована для WEB–розробок, і її код може впроваджуватися

безпосередньо в HTML. PHP входить до десятки найпопулярніших мов програмування. На ньому написано більшість CMS (систем управління контентом), у тому числі WordPress, на якому працює 65% всіх сайтів у світі. Розробники розвивають мову вже понад 25 років та впроваджують нові можливості з кожною версією. Щоб підтримувати роботу старих проектів та писати WEB–програми з нуля, компанії наймають програмістів на PHP [11].

В якості головного інструмента для оптимізації та обслуговування бази даних (БД) буде використовуватись мова програмування SQL. Structured Query Language – це мова програмування структурованих запитів, яка використовується як ефективний спосіб збереження даних, пошуку, оновлення, витягування або видалення з бази даних. За допомогою SQL можна виконувати наступний набір операцій:

- створення у базі даних нових таблиць;
- додавання до таблиці нових записів;
- зміна записів;
- видалення записів;
- вибірка записів з однієї або декількох таблиць;
- зміна структур таблиць тощо.

3.2.3. Вибір рішення для серверної частини

Для вирішення задач авторизації, реєстрації та взаємодії між WEB–додатком та сервером використаємо фреймворк Laravel. Фреймворк – це каркас, який складається з багатьох бібліотек, які полегшує розробку продукту. Фреймворк відрізняється від поняття бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підпрограм близької функціональності, не впливаючи на архітектуру WEB–додатку і не накладаючи на неї жодних обмежень [12].

Laravel є безкоштовним PHP фреймворком загального призначення з відкритим кодом. Він написаний мовою програмування PHP і створювати код

на його базі можна також тільки на PHP. Сьогодні Laravel є одним з найпопулярніших движків. З його допомогою можна однією командою згенерувати систему реєстрації та входу у WEB–додаток та з легкістю підключити сервіси OAuth для можливості входу у профіль через соціальні мережі [12].

3.3. Реалізація графічного інтерфейсу WEB–додатку та його компонентів

3.3.1. Створення логотипу

Для того, щоб створити логотип, що запам’ятовується, спочатку потрібно вигадати назву для WEB–додатку. Основне призначення додатку – це надання послуг особистого каршерінгу. Переведемо його англійською мовою для майбутнього виходу WEB–додатку на міжнародний рівень – *«personal carsharing services».* Така назва є занадто довгою, отже, необхідно скоротити – *«your own carsharing».*

Для створення логотипу використаємо абревіатуру «YOCar», обрана кольорова гамма буде складатись з темно зеленого та бежевого кольорі, а так як основним направленням додатку є здача машин в оренду, то головним символом буде саме машина.

За допомогою графічного редактору Photoshop було створено декілька варіантів логотипу та обрано найкращий, що зображений на рис. 3.1.

Рис. 3.1. Логотип WEB–додатку власного каршерінгу

3.3.2. Створення інтерфейсу сторінок входу та реєстрації

Першим етапом розробки WEB–додатку є створення сторінок авторизації та реєстрації користувача. Коли користувач зайшов у WEB–додаток YOCar, йому має надаватись вибір функцій: або зареєструватися, якщо користувач вперше зайшов у WEB–додаток, або авторизуватись, якщо профіль був уже створений. Перша сторінка зображена на рис. 3.2.

Рис. 3.2. Початкова сторінка WEB–додатку власного каршерінгу

При натисканні на кнопку буде відбуватись перехід на відповідну сторінку. Сторінки входу та реєстрації зображені на рис. 3.3 – 3.4.

Рис. 3.3. Сторінка авторизації у WEB–додатку

Якщо користувач забувся пароля, він може звернутись в підтримку, натиснувши на текст «Забули пароль?», що розміщений під полями вводу для подальшого його відновлення.

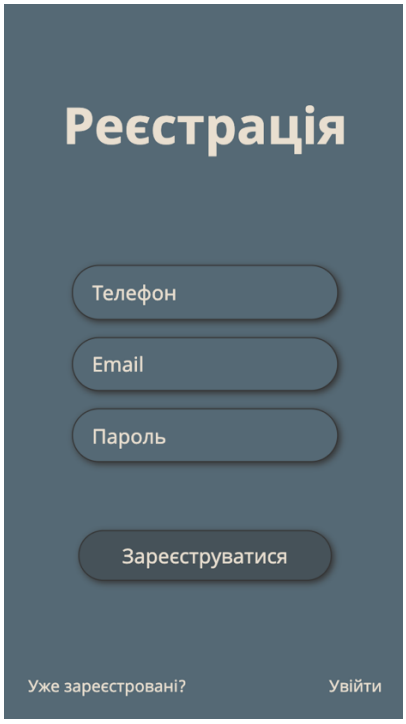

Рис. 3.4. Сторінка реєстрації у WEB–додатку

Поля для реєстрації, такі як: телефон, email та пароль, представлені у вигляді HTML–форми для взаємодії між користувачем та WEB–додатком.

```
<form action="" id="modal-auth-form" class="flax-column">
<input type="text" class="input-form" placeholder="Телефон" name="phone">
<input type="email" class="input-form" placeholder="Email" name="login">
<input type="password" placeholder="Пароль" name="password" >
<input id="form-signup-btn" type="submit" value="Зареєструватися">
</form>
```

Інформація HTML–форм передається за допомогою елемента <input>. Для введення одного рядка тексту застосовується елемент <input type="text">, для введення електронної адреси – <input type="email">, а для введення пароля, що буде відображатись зірочками, – <input type="password">. Дані введені користувачем відправляються на сервер за допомогою кнопки <input type="submit"> для подальшої обробки та зберігання.

Якщо користувач ввів некоректно дані, то буде відображатися підказка (Рис.3.5) для подальшої точної обробки цих даних.

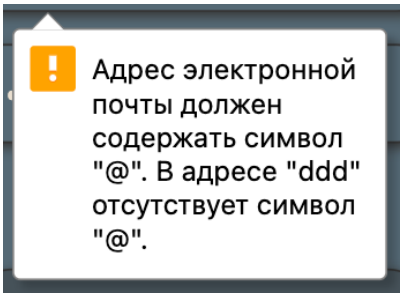

Рис. 3.5. Відображення підказки при заповнені форми

Після введення даних в поля для реєстрації та натискання кнопки «Реєстрація», користувачу на вказаний номер надійде код підтвердження, який треба ввести у відповідне поле, що зображено на рис. 3.6, та натиснути «Підтвердити».

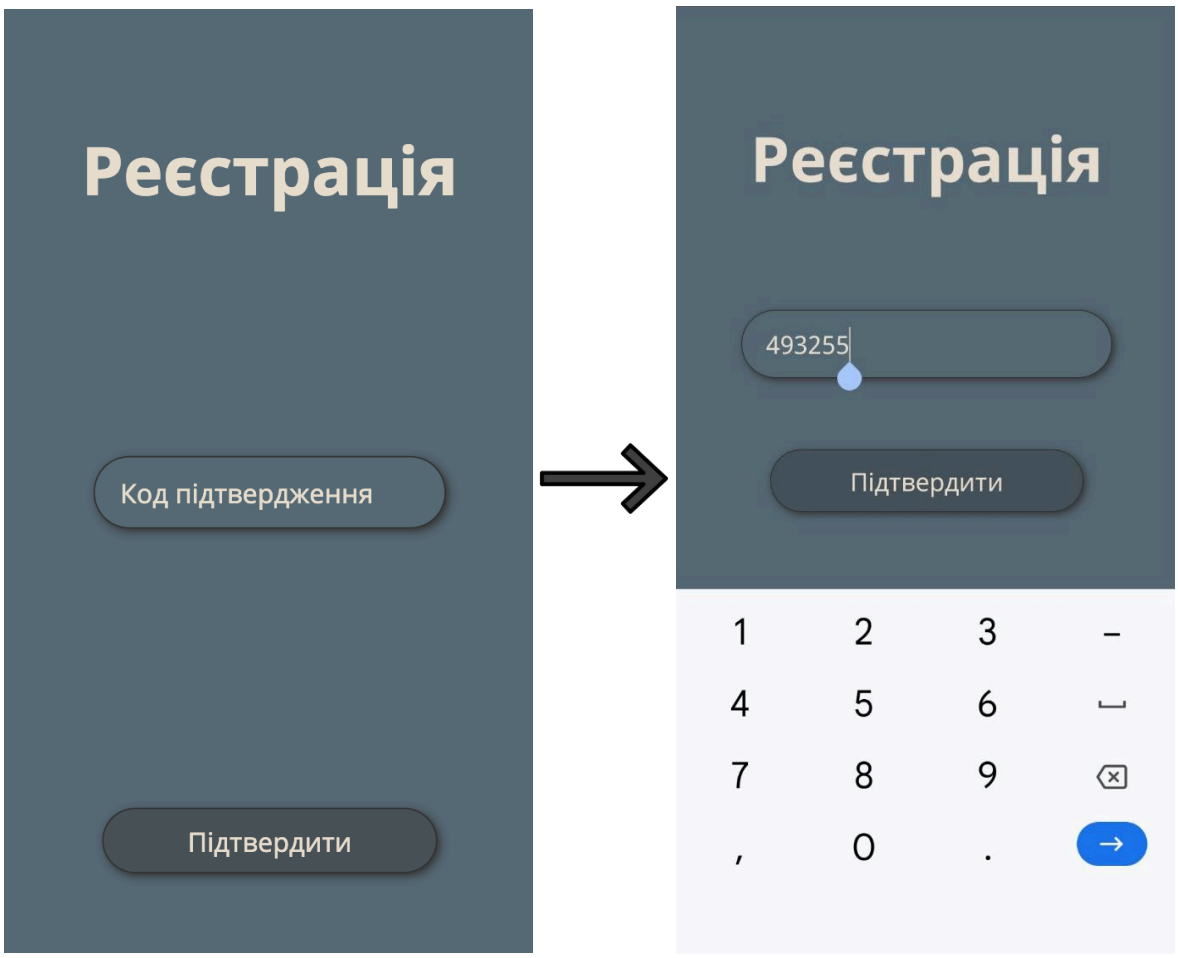

Рис. 3.6. Поле підтвердження номеру телефону

Після введення вірного коду відбувається перехід на головну сторінку WEB–додатку.

3.3.3. Створення інтерфейсу головної сторінки

Головна сторінка WEB–додатку YOCar повинна містити навігацію до будь-якої іншої сторінки. Саме тому зверху екрану буде розміщене випадаюче меню. Зверху повинні бути доступні кнопки фільтру та перегляду доступних автомобілів на мапі, а сітка автомобілів має бути зручною для прокручування та порівнювання запропонованих варіантів. На рис. 3.7. зображена головна сторінка WEB–додатку YOCar.

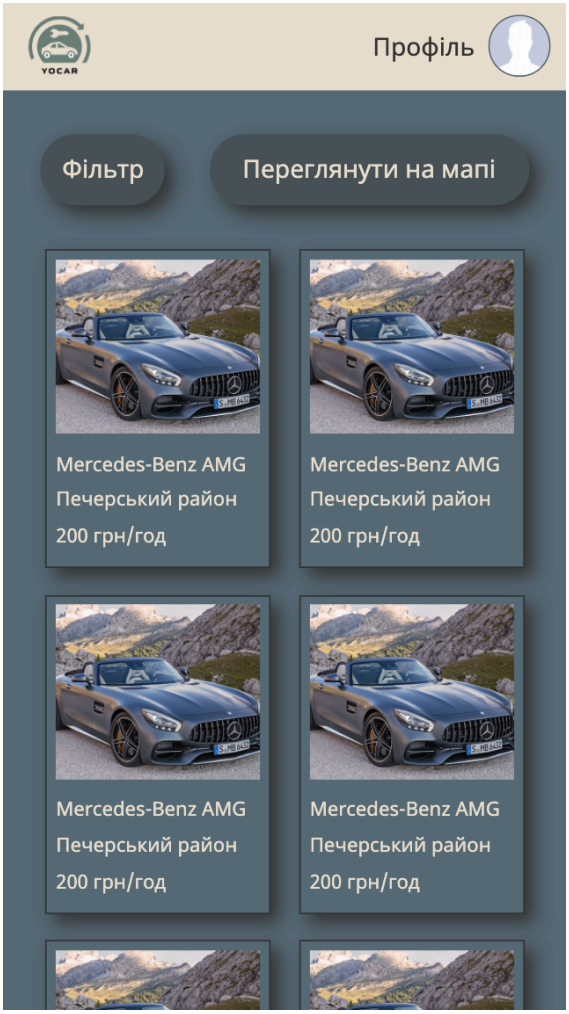

Рис. 3.7. Головна сторінка WEB–додатку

Якщо прогорнути сторінку донизу, то з’явиться кнопка «Більше» (рис.3.8), що завантажить більше варіантів для вибору.

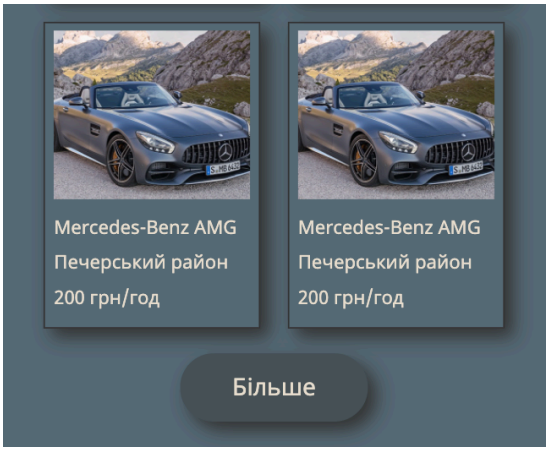

Рис. 3.8. Кнопка «Більше» на головному екрані

Випадаюче меню профілю, що зображене на рис. 3.9, містить такі пункти як: запити, редагування профілю, перегляд профілю та вихід з профілю. Поява цього меню відбувається за рахунок використання JavaScript. До блоку, що має стиль відображення display: none, при натисканні додається новий клас, який змінює відображення цього блоку на display: flex.

Рис. 3.9. Відкриття випадаючого меню

3.3.4. Створення інтерфейсу сторінки фільтру

Сторінка фільтрів відкривається при натисканні відповідної кнопки на головній сторінці. Відфільтрувати вміст головної сторінки можна за наступними параметрами:

- клас авто;
- вартість;
- тип пального;
- тип трансмісії;
- маркер вільно;
- розташування.

Для того щоб створити повзунок ціни, спочатку потрібно знайти повзунок з схожим функціоналом в Інтернеті, скачати його та перемісти в необхідну папку для подальшого використовування. Наступним кроком необхідно підключити скрипт перед </body> за допомогою наступного коду:

<script src="src/rSlider.js"></script>

Для того щоб відобразити повзунок на сторінці, необхідно створити новий об’єкт, в якому прив’язати повзунок до id, задати мінімальне та максимальне значення повзунку, тип повзунка, параметри початкового відображення мінімальної та максимальної ціни на повзунку, крок зміщення при перетягуванні повзунка та стилі.

```
var mySlider = new rSlider({
target: '#slider',
values: {min:50, max:800},
range: true, // range slider
step: 1,
set: [200, 400],
scale: false,
labels: false,
tooltip: false,
});
```

Для того щоб користувач міг бачити, який діапазон ціни він обрав, необхідно передати значення в input, створивши наступну функцію:

```
onChange: function(v){
var arrprice = v.split(',');
$('#min-price').val(arrprice[0]);
$('#max-price').val(arrprice[1]);
}
```

Повзунок фільтру ціни зображений на рис. 3.10.

Рис. 3.10. Повзунок фільтру ціни

Далі необхідно створити фільтри із чекбоксами, щоб у користувача була можливість зручно фільтрувати відображення авто. Кожен чекбокс треба помістити в окремий div для майбутньої зручної стилізації. А для того, щоб знати, яке саме значення буде передаватись тегу <label></label>, треба задати атрибут for, в який вписати id тега <input/>

<input type="checkbox" id="mini-cars" name="scales"/>

<label for="mini-cars">Мікроавтомобілі</label>

Повна сторінка фільтрів зображена на рис. 3.11.

Рис. 3.11. Сторінка фільтрів

3.3.5. Створення інтерфейсу сторінки мапи

Одним з найважчих етапів створення WEB–додатку є підключення Карти Google, так як це займає досить великий проміжок часу для розуміння документації.

Карти Google – це комплекс програм, створених на базі безкоштовного сервісу картографії, що використовується Google. Набір додатків Карт Google надає можливість розміщувати на своєму WEB–сайті:

- мапу;
- маршрут проїзду;
- результати пошуку інформації на карті з відмітками пам'яток або організацій;
- панорами з перегляду вулиць.

Щоб створити карту Google із маркерами на WEB–сторінці, потрібно виконати наступні кроки [13]:

1. Створення сторінки WEB–додатку. Сторінка може містити будь-який вміст, але в моєму випадку там буде розміщена лише мапа з маркерами та можливістю переходу на сторінку карточки обраного авто. На рис. 3.12 зображений процес створення PHP файлу, що реалізується як динамічна WEB–сторінка з інтерактивними функціями. PHP двигун, який встановлений у WEB–сервер аналізує PHP коди, збережені у файлі PHP, і відповідні HTML-дані динамічно відображається для глядача.

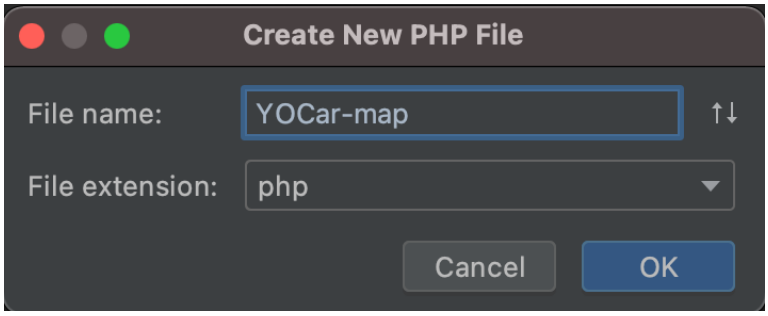

Рис. 3.12. Створення PHP файлу

2. Визначення області сторінки для карти Google, створивши блок div з заданими для нього висотою та шириною.

```
<div id="contact-map" class="contact-map"></div>
.contact-map{
display:block;
width:100%;
height:100%;
}
```

3. Підключення скрипту із зовнішнього файлу із розширенням .js перед тегом </body>,Скрипт буде містить посилання на програму або її текст певною мовою, в моєму випадку це JS.

```
<script src="src/script.js?v=3.1"></script>
```

Цей скрипт може зв'язуватися з будь-яким HTML–документом. Такий підхід дозволяє використовувати ті самі спільні функції на багатьох WEB–сторінках, прискорюючи їх завантаження, так як. зовнішній файл кешується при першому завантаженні, і скрипт викликається швидше при наступних викликах.

4. Завантаження Maps JavaScript API на WEB–сторінку за допомогою власного ключа API, написавши в <body> код наведений нижче:

```
<script src="https://maps.googleapis.com/maps/api/js?key=AI.. "></script>
```

Параметр містить ключ API, який можна купити.

5. Написання власного JavaScript, який використовує API для додавання карти з маркером на ньому. Для цього створюється функція, в якій створюються координати центру мапи, задається сила зуму та стиль самої мапи, який обирається в безкоштовних ресурсах. Також було вирішено вимкнути стандартні кнопки на мапі, щоб користувачу нічого не заважало керувати нею [15].

```
function initGmapsContacts() {
var coordinateCenter = {lat: 50.4128887, lng: 30.5325584};
```

```
const map = new google.maps.Map(document.querySelector('#contact-
map'), {
center: coordinateCenter,
zoom: 15,
disableDefaultUI: true,
styles:…
});
```

На рис. 3.13 зображений результат написаного коду:

Рис. 3.13. Відображення стилів мапи у WEB–додатку

6. Наступним кроком необхідно створити маркер, який буде вказувати на розміщення вільної автівки на мапі. Це реалізується за допомогою створення нового об’єкту, що має критерії розташування, виводу тексту при наведенні та візуального зображення.

```
const marker = new google.maps.Marker({
position: coordinateCenter,
map,
```

```
title: "car1",
icon: image,
});
```

Стандартний маркер було вирішено змінити для більш приємного візуального відображення зв’язаного з моєю темою дипломної роботи. Це було реалізовано за допомогою додавання url зображення та коректного позиціонування його на мапі.

```
var image = {
url: "/src/images/car-map.svg",
size: new google.maps.Size(91, 91),
origin: new google.maps.Point(-20, 0),
anchor: new google.maps.Point(0, 0),
scaledSize: new google.maps.Size(50, 50)
};
```

Результатом даного кроку є відображення маркеру на мапі (рис.3.14).

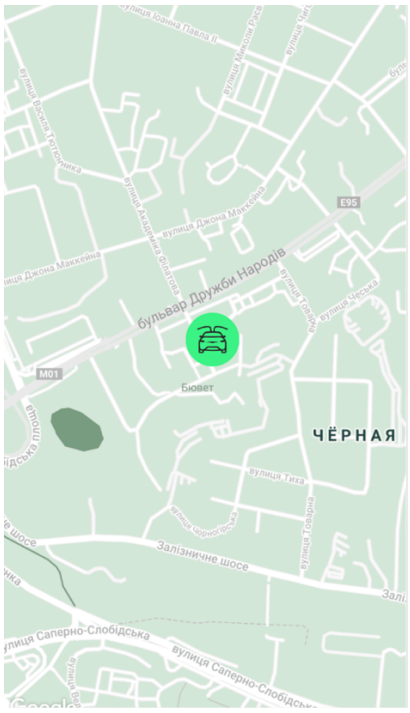

Рис. 3.14. Відображення маркеру на мапі у WEB–додатку

7. Додавання спливаючого вікна при натисканні на маркер необхідно для зручної навігації користувача, так як перед тим як прокласти маршрут до точки, користувачу необхідно переглянути обрану автівку, перейшовши на сторінку з деталями та інформацією про оренду [14]. Для цього необхідно додати подію при натисканні на маркер, за допомогою наступного уривку коду:

```
marker.addListener("click", () => {
infoWindow.open({
anchor:marker,
map,
shouldFocus: false,
});
});
```

Для відображення інформації після натискання необхідно її додати:

```
const contentCar = '<a class="btn-view-car" href="/carinfo.php">
Переглянути</a>';
const infoWindow = new google.maps.InfoWindow({
content: contentCar,
mawWidth: 200,
});
```

Результатом, який зображений на рис. 3.15, є відображення кнопки, за допомогою якої буде здійснюватися перехід на сторінку картки автомобіля.

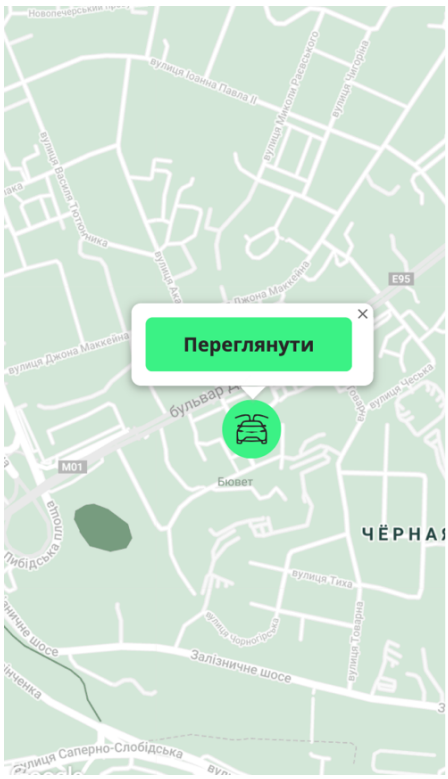

Рис. 3.15. Відображення спливаючого вікна при натисканні на маркер

Для подальшого відображення на мапі маркерів всіх авто з БД, було написано обробник на мові програмування JS для створення маркерів кожного з автомобілів та зв’язування даних з бекенду та фронтенду. За допомогою поєднання мов PHP та JS було створено масиви, що складаються з інформації, яка спливає при натисканні, точок що відображаються на карті та події слухання кліку на кожен з маркерів.

Також було додано ще одну кнопку до спливаючого вікна при натисканні, за допомогою якої відбувається перехід в Google Maps з передачею координат обраного авто для подальшого прокладання маршруту до точки:

coord1}},{{$car- >coord2}}">

Іконка маркеру авто може мати статус зарезервовано або вільно і за допомогою перевірки статусу можлива зміна кольору іконки на мапі.

Відображення масиву вільних та зарезервованих авто на мапі з можливістю переходу на сторінку Google Maps для прокладання маршруту демонструється на рис. 3.16.

Рис. 3.16. Відображення масиву всіх авто

3.3.6. Створення інтерфейсу сторінки власного профілю

Будь-який авторизований користувач матиме змогу переглядати свій профіль та профіль будь-якого іншого користувача WEB–додатку. Профілі можна класифікувати за двома типами:

- профіль користувача, що здає автомобіль в оренду (рис. 3.17);
- профіль користувача, що планує користуватися додатком лише з ціллю оренди (рис. 3.18);.

На відміну від другого, перший має на своєму профілі не лише інформацію про себе, а й всі необхідні дані про автомобіль: фото авто, повний опис, маркер «вільно», що змінює дизайн при натисканні за допомогою JS та кнопку оновлення розташування авто на мапі.

Також кожен профіль має в кінці сторінки список відгуків з можливістю залишення їх іншими користувачами.

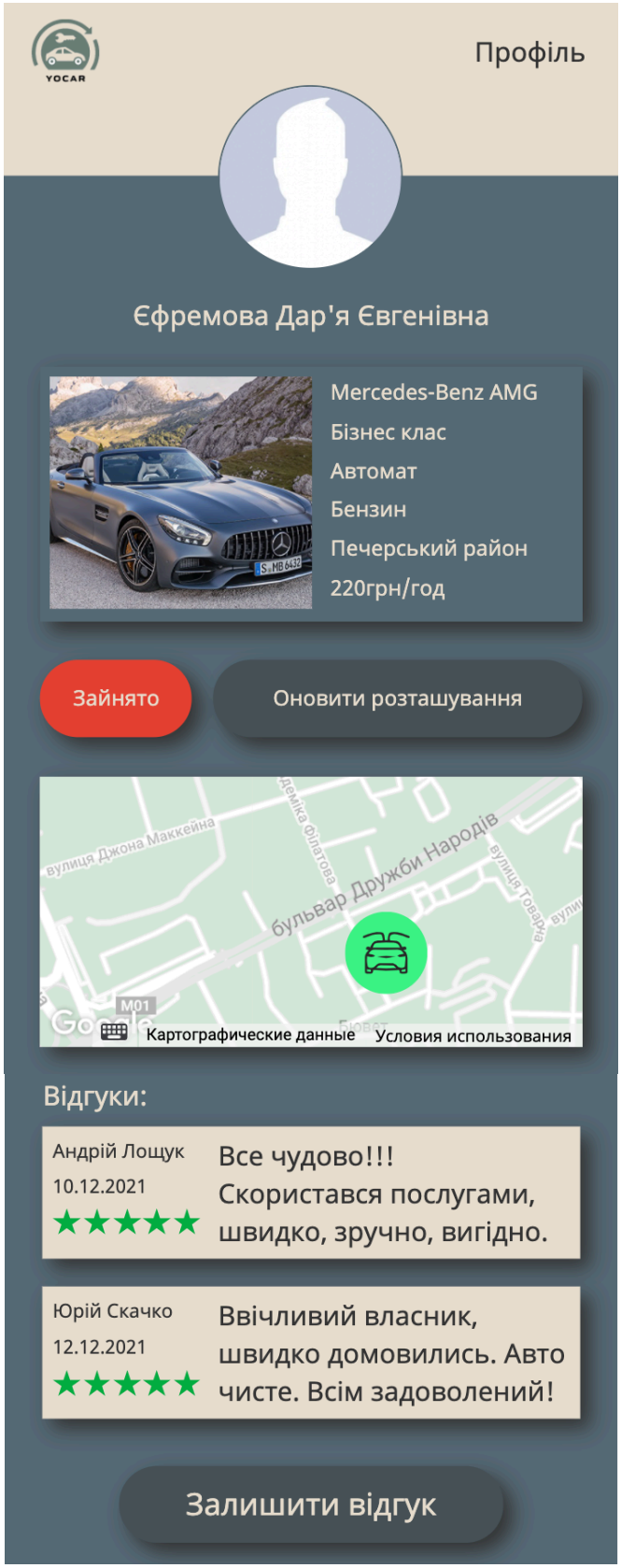

Рис. 3.17. Інтерфейс сторінки профілю автовласника

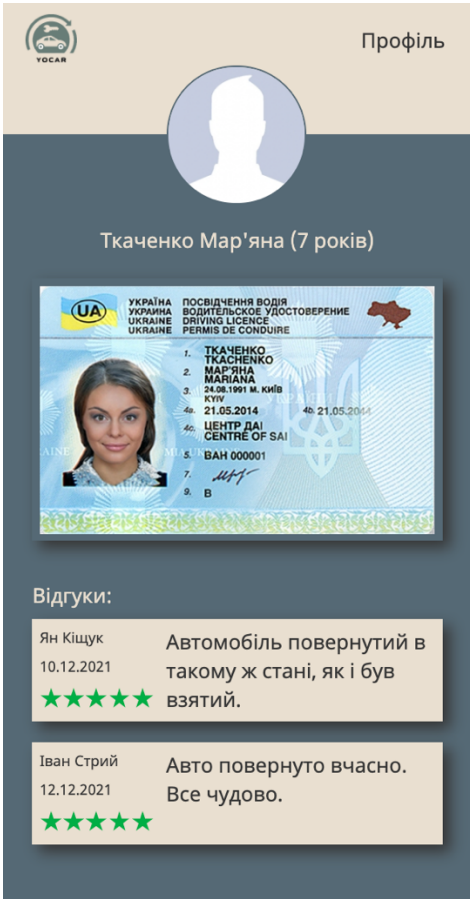

Рис. 3.18. Інтерфейс сторінки профілю орендаря

3.3.7. Створення інтерфейсу сторінки картки авто

Інтерфейс сторінки картки авто, що зображений на рис. 3.19, схожий з інтерфейсом сторінки профіля користувача, авто якого переглядається в даний момент, але знизу ще доданий функціонал, що надає можливість обрати дату та час оренди для відправлення запиту, що реалізується за допомогою форми:

<form action="">

<input type="date" name="date">

<input type="time" name="time-from">

<input type="time" name=" time-to ">

<input id="form-signup-btn" type="submit" value="Залишити запит">

</form>,

де <input type="date"> слугує для вводу дати оренди, а <input type="time"> – для вводу часу.

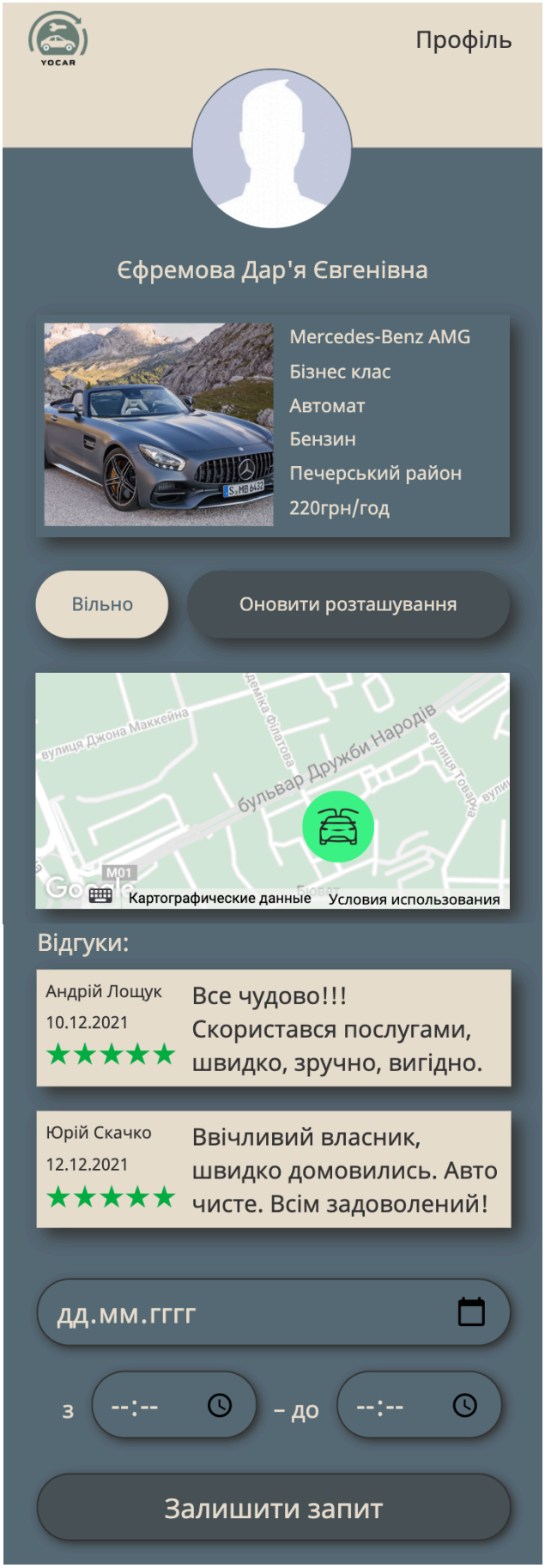

Рис. 3.19. Інтерфейс сторінки картки авто

3.3.8. Створення інтерфейсу сторінки перегляду запитів

Дана сторінка існує для зручного перегляду отриманих запитів. Фото орендаря, дата та час запрошуваної поїздки відображається в полі запиту. При натисканні на нього, відбувається перехід на сторінку орендаря для того, щоб орендодавець зміг подивитись досвід водіння та відгуки, на основі яких прийняти рішення: підтвердити оренду чи відмовити. Подвійне натискання на поле підтвердження – змінює статус поля на «Запит відхилено», подвійне – «Запит прийнято». Відповідно до прийнятого рішення буде змінюватися SVG–зображення (хрестик або галочка).

Інтерфейс сторінки перегляду запитів зображений на рис. 3.20.

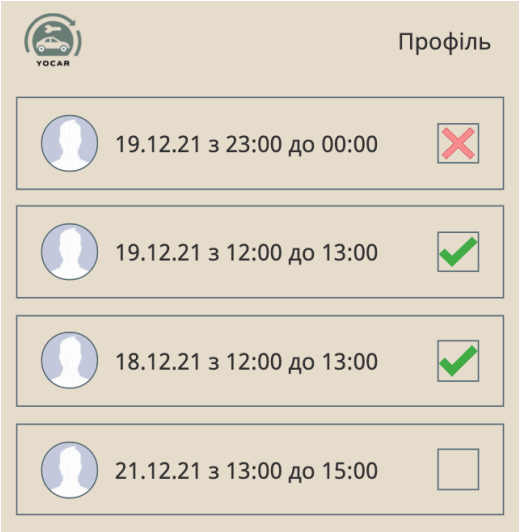

Рис. 3.20. Інтерфейс сторінки перегляду запитів

Коли орендодавець визначився з рішенням, користувачу прийде сповіщення у вигляді спливаючого вікна. Натиснувши на відповідну кнопку, відбудеться зв’язок орендаря та орендодавця в месенджері.

Інтерфейс спливаючого вікна про підтвердження запиту зображений на рис. 3.21.

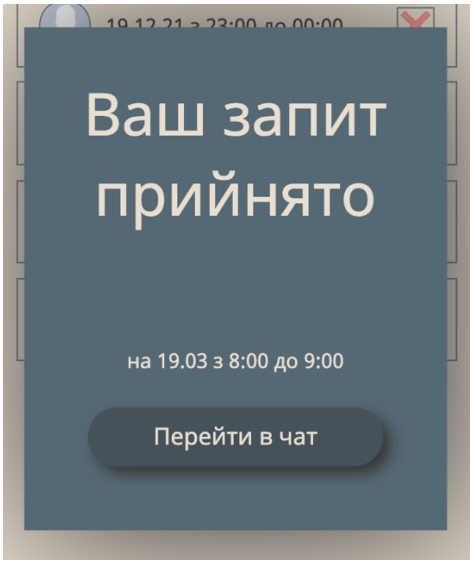

Рис. 3.21. Інтерфейс спливаючого вікна про підтвердження запиту

3.3.9. Створення інтерфейсу сторінки редагування або додавання інформації в профіль

Вигляд сторінки додавання інформації та її редагування зображений на рис. 3.22.

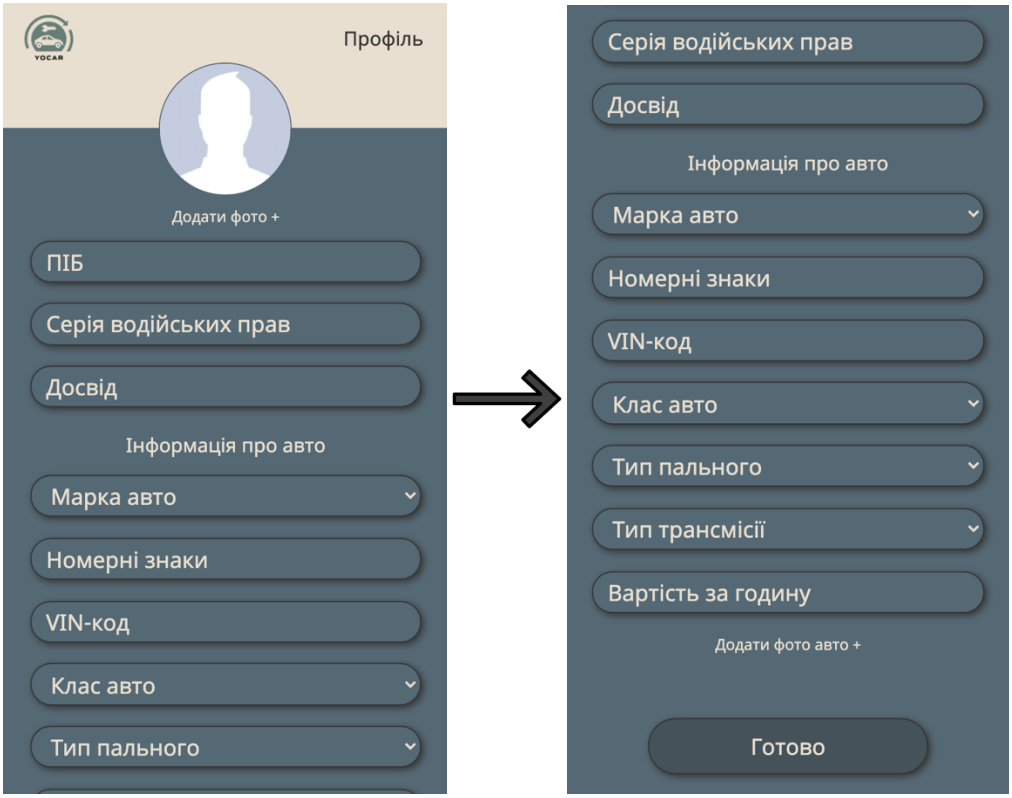

Рис. 3.22. Інтерфейс сторінки редагування профілю

Поля для заповнення інформації створені за допомогою <input type="text"> та <select> </select>. Другий дозволяє створити елемент у вигляді списку, що розгортається (рис. 3.23).

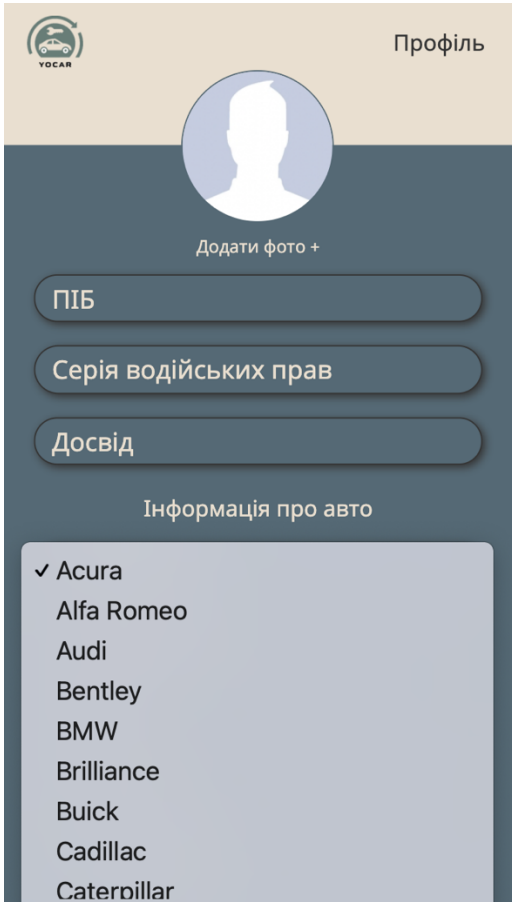

Рис. 3.23. Вибір марки авто

3.4. Реалізація і налаштування серверної частини

Для реалізації серверної частини для початку необхідно встановити Laravel та репозиторій за допомогою терміналу для того, щоб було зручно створювати базу даних та щоб був доступ адміністратора. Для того щоб створити та керувати базою даних, необхідно запустити локальний WEB–сервер XAMPP, в якому міститься необхідна програма. Процес створення бази даних зображений на рис. 3.24.

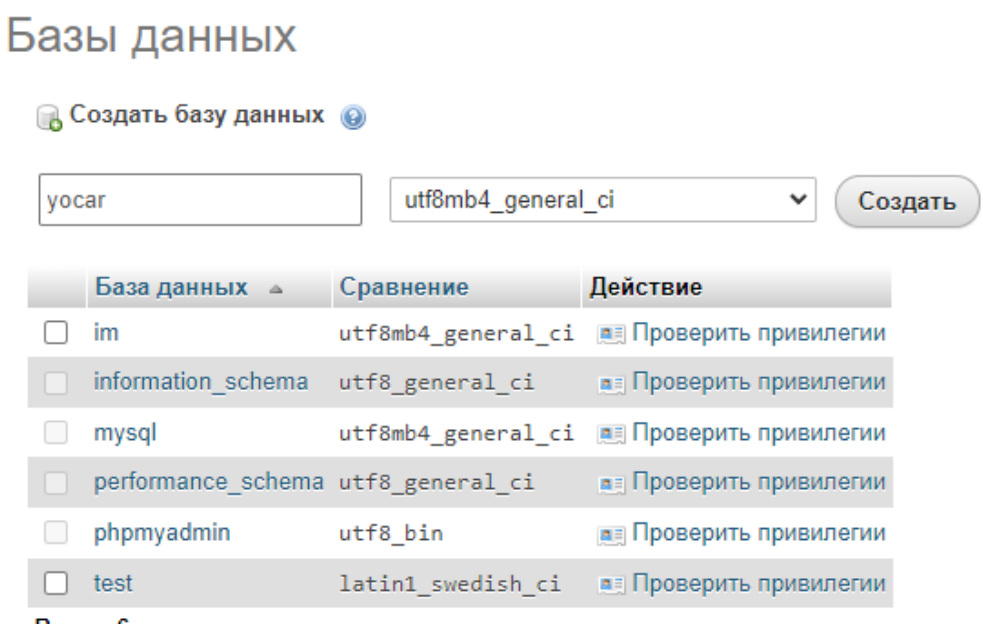

Рис. 3.24. Створення бази даних

Необхідно підключити Laravel до БД, змінивши файл з налаштуваннями (рис. 3.25).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=yocar
DB_USERNAME=root
DB_PASSWORD=
```

Рис. 3.25. Підключення Laravel

В IDE PhpStorm через термінал було підключено систему, яка емулює CMS та CRUD, зареєстровано себе як адміністратора та в режимі тестування обрано одну мову (рис. 3.26).

```
PS D:\xampp2\htdocs\YOCar> php artisan fastadminpanel:install
Please note: FastAdminPanel requires fresh Laravel installation!

 Languages count:
 > 1

 ID of main language from 0 to 0:
 > 0

 Language tag number 0:
 > uk

If you missclicked with something, you can repair it in table languages
Single tables created
DB creation complete!

 Administrator name:
 > Dasha

 Administrator email:
 > dashaa_efremova@icloud.com

 Administrator password:
 >

User has been created
Menu has been created
```

Рис. 3.26. Підключення системи емуляції

На рис. 3.27 зображено налаштування локального домену.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:qnX/+P4fbZEJWs7Nw
APP_DEBUG=true
APP_URL=http://yocar.local
```

Рис. 3.27. Налаштування локального домену

Необхідно виконати перевірку успішного виконання всіх кроків. Для цього необхідно запустити сторінку. При успішному запуску на екран має вивестись текст «HW!» (рис. 3.28).

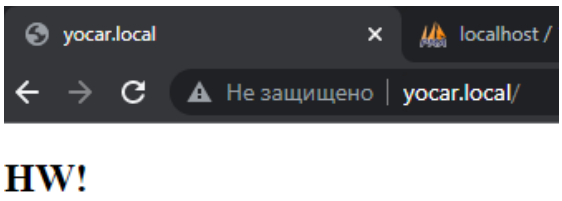

Рис. 3.28. Успішний запуск локального домену

Якщо в командному рядку дописати «/admin», здійсниться вхід в панель адміністратора, де в подальшому можна відстежувати дані користувачів в БД. Вхід в панель адміністратора зображено на рис. 3.29, а на рис. 3.30 зображений стандартний вигляд панелі адміністратора.

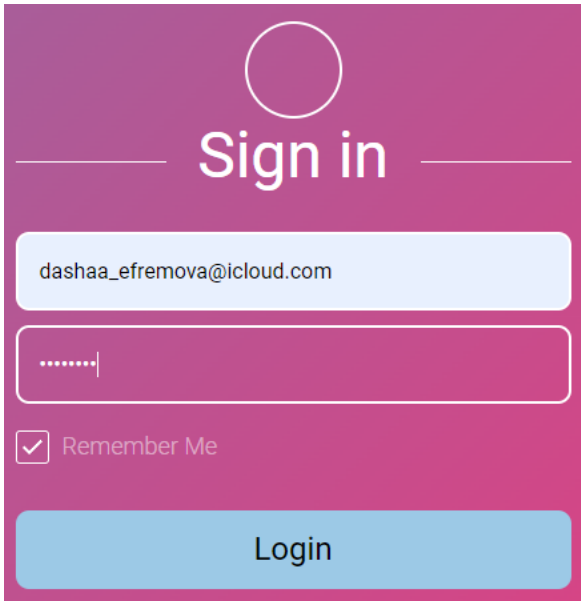

Рис. 3.29. Вхід в панель адміністратора

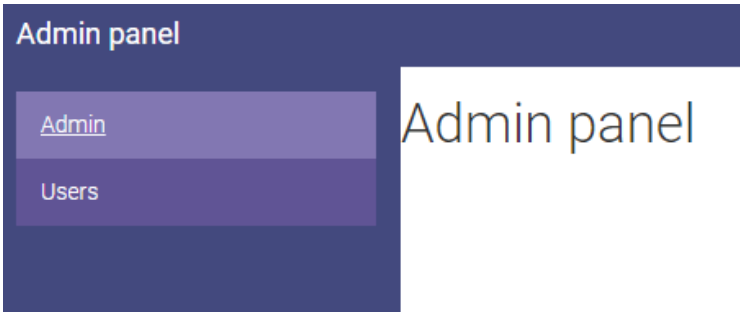

Рис. 3.30. Стандартний вигляд панелі адміністратора

За допомогою скритого функціоналу панелі адміністратора можна створити таблиці в БД (рис. 3.31).

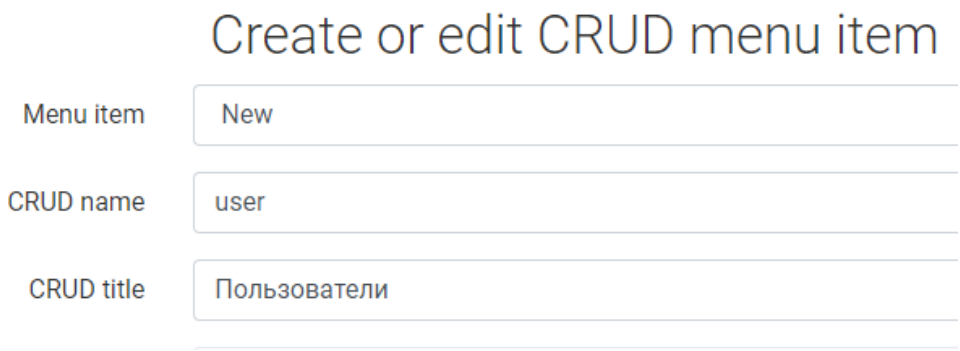

Рис. 3.31. Створення нової таблиці

Всього необхідно створити чотири таблиці:

– таблиця, що зберігає всі необхідні дані про користувача (рис. 3.32)
– таблиця, що зберігає всі необхідні дані про автомобіль (рис. 3.33)
– таблиця, що зберігає всі необхідні дані про запити (рис. 3.34)
– таблиця, що зберігає всі необхідні дані про відгуки (рис. 3.35)

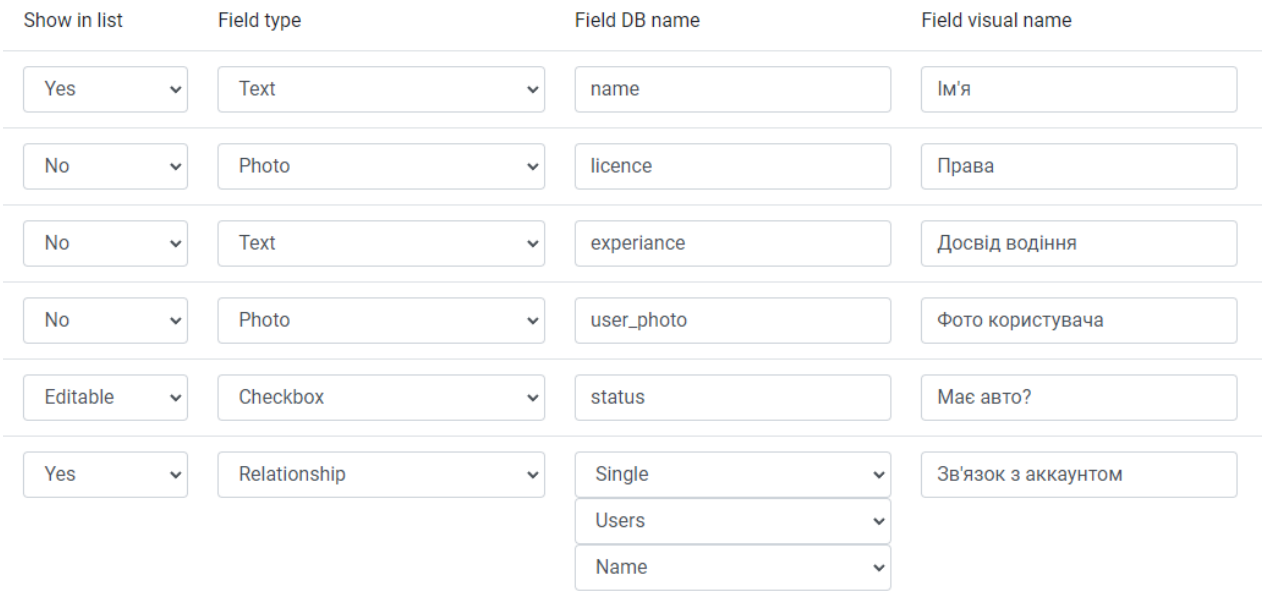

Рис. 3.32. Таблиця Users

Show in list	Field type	Field DB name	Field visual name
Yes	Text	mark	Марка
Yes	Text	num	Номер авто
No	Text	vin	Vin-код
No	Text	class	Клас авто
No	Text	fuel	Тип пального
No	Text	trans	Тип трансмісії
Yes	Money	price	Ціна
No	Photo	main_photo	Фото авто
No	Gallery	gallery	Галерея авто
Yes	Relationship	Single Users Name	Власник
No	Text	coord1	Координата1
No	Text	coord2	Координата 2
Editable	Checkbox	status	Статус

Рис. 3.33. Таблиця Cars

Show in list	Field type	Field DB name	Field visual name
Yes	Relationship	Single Users Name	Від кого
Yes	Relationship	Single Авто Номер авто	До кого
Yes	Date and time picker	time_from	Час початку
Yes	Date and time picker	time_to	Час завершення
Editable	Checkbox	status	Статус підтвердження

Рис. 3.34. Таблиця Requests

Show in list	Field type	Field DB name	Field visual name
Yes	Relationship	Single Users Name	Кому
Yes	Text	name	Від кого
Yes	Number	mark	Оцінка
No	Long text	text	Текст

Рис. 3.35. Таблиця Responses

Вигляд панелі адміністратора після створення таблиць (рис. 3.36).

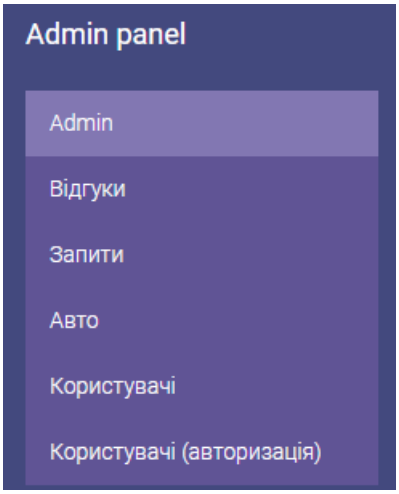

Рис. 3.36. Оновлена панель адміністратора

Для подальшого налаштування бекенду, необхідно перенести написаний фронтенд в фреймворк Laravel.

Першим кроком в створенні бекенду – є створення контролерів для груп схожих сутностей та написання коду для кожної сторінки (рис. 3.37), в якому задається метод, майбутня назва URL сторінки та присвоєно виконання майбутньої функції на відповідній сторінці.

Контролер необхідний для керування бекендом, адже в ньому описуються функції, за допомогою яких існує зв'язок сутностей з бази даних і фронтенду.

```
Route::get( uri: '/catalog', action: 'CatalogController@catalog');
Route::get( uri: '/carinfo/{id}', action: 'CatalogController@carinfo');
Route::get( uri: '/filter', action: 'CatalogController@filter');
Route::get( uri: '/map', action: 'CatalogController@map');
```

Рис. 3.37. Приклад групи схожих сутностей

Для того, щоб контент відобразився на сторінці, необхідно в відповідному контролері створити функцію, що повертає відображення контенту (рис. 3.38).

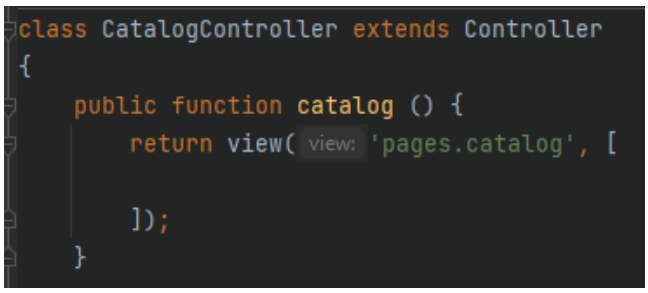

Рис. 3.38. Функція для відображення сторінки каталогу авто

На рис. 3.39 зображена перевірка авторизації користувача при вході у WEB–додаток. Якщо користувач раніше зайшов в свій профіль з свого пристрою, він перенаправляється на сторінку каталогу авто, якщо ні – користувачу відображається перша сторінка з вибором: авторизація або реєстрація.

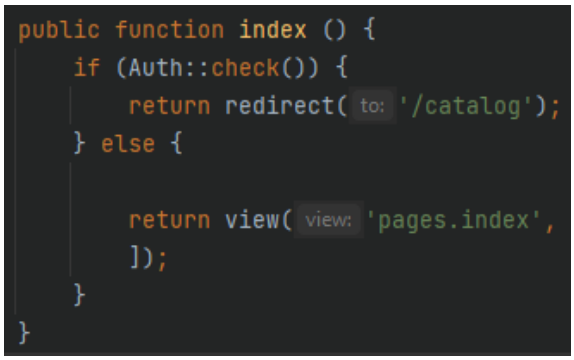

Рис. 3.39. Перевірка авторизації

На рис. 3.40 зображена функція за допомогою якої відбувається вихід з WEB–додатку.

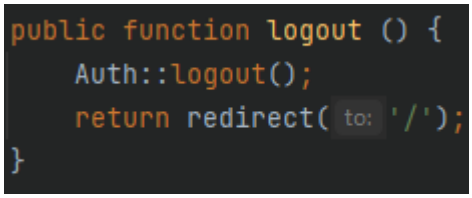

Рис. 3.40. Вихід з WEB–додатку

Для того щоб правильно виконувалась функція авторизації, в першу чергу необхідно задати метод Post в формі для вводу даних авторизації. Наступним кроком виконується перевірка, чи авторизований користувач. Якщо так – відбувається перехід на сторінку каталогу, якщо ні – то після введення в форму даних, на сервер надсилається два поля (логін та пароль). Далі відбувається перевірка наявності в базі даних введеного логіну. Якщо він присутній, то відбувається порівнювання надісланого захешованого паролю з наявним в базі даних захешованим паролем. Якщо всі введені користувачем дані відповідають даним в БД, то авторизація пройшла успішно і відбувається перехід на сторінку каталогу.

Уривок коду реалізації процесу авторизації зображено на рис. 3.41.

```
public function authorization () {
    if (Auth::check()) {
        return redirect( to: '/catalog');
    } else {
        if(isset($_POST)&&!empty($_POST)){
            $login = $_POST['login'];
            $password = $_POST['password'];
            $user = User::where('email', $login)
                ->first();

            if ($user != null && Hash::check($password, $user->password)) {
                Auth::login($user);
                return redirect( to: '/catalog');
            } else return redirect( to: '/authorization');
        }
        return view( view: 'pages.authorization', [

        ]);
    }
}
```

Рис. 3.41. Реалізація процесу авторизації

Для реєстрації користувача виконується перевірка даних на їх коректність (рис. 3.42). За допомогою вбудованого в Laravel класу Validator викликається функція make, яка порівнює дані на вході з заданими шаблонами

та умовами, наприклад: поле має бути обов’язково заповнено, дані, що передаються, повинні бути унікальними, а максимальна кількість символів не повинна перевищувати задане значення.

```php
if(isset($_POST)&&!empty($_POST)){
    $validator = Validator::make($_POST, [
        'email'     => [
            'required',
            'email',
            'unique:users,email',
            'max:191',
        ],
        'phone'     => [
            'required',
            'unique:users,phone',
            'regex:/\+{0,1}\d{4,40}/',
        ],
        'password'  => [
            'required',
            'max:60',
        ],
    ]);

    if ($validator->fails()) {
        return $this->error($validator->errors());
    }
```

Рис. 3.42. Перевірка даних при реєстрації

Для того щоб зберегти нового користувача WEB–додатку в БД, виконується підтвердження введених даних, шляхом надсилання коду підтвердження. Реалізація даного процесу відбувається наступним чином: користувач створюється у БД (рис. 3.43), йому присвоюється код підтвердження, що рандомно генерується за допомогою функції generateRandomString (рис. 3.44), а вже згенерований код надсилається користувачу.

Рис. 3.43. Функція створення користувача

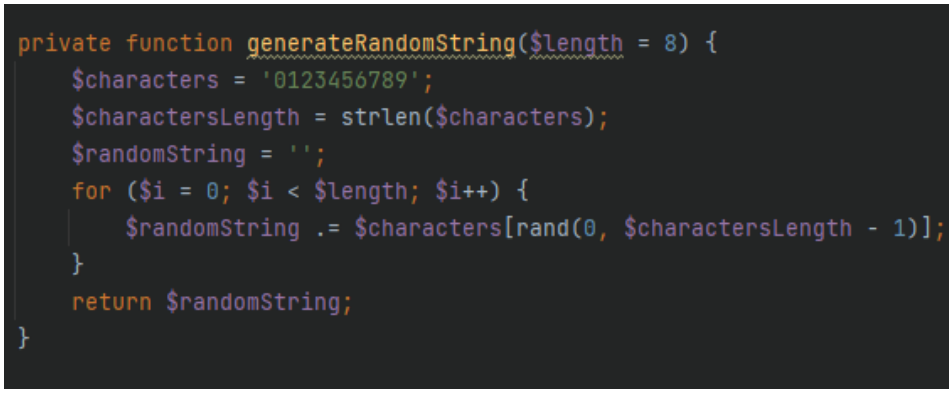

Рис. 3.44. Генерація випадкового коду

Для підтвердження реєстрації відбувається порівняння збереженого в БД згенерованого коду з тим, що ввів користувач в відповідне поле. Якщо користувача з введеним кодом не існує в БД – користувач лишається на тій самій сторінці, якщо існує – то відбувається оновлення таблиці, а саме: статус користувача змінюється на авторизований, а код підтвердження видаляється з БД. Після успішної реєстрації відбувається перехід на сторінку каталогу авто, якщо ні.

На рис. 3.45 зображений уривок коду підтвердження реєстрації.

```php
public function registrationCode () {
    if (Auth::check()) {
        return redirect( to: '/catalog');
    } else {
        if(isset($_POST)&&!empty($_POST)){
            $user = User::where('code',$_POST['code'])
                ->first();
            if($user) {
                Auth::login($user);
                User::where('code', $_POST['code'])
                    ->update(['code' => '','is_active'=>1]);
                return redirect( to: '/catalog');
            } else {
                return redirect( to: '/code');
            }
        }
        return view( view: 'pages.registrationCode', [

        ]);
    }
}
```

Рис. 3.45. Підтвердження коду

На рис. 3.46 зображений уривок коду, за допомогою якого відбувається вивід даних користувача у відповідні поля на сторінці *редагування*, шляхом отримання і передачі даних на неї.

Замість прямих запитів в SQL використовується PDO Laravel [16].

```php
$user = [];
$userdata = [];
$cardata = [];
if(Auth::check()) {
    $user = Auth::user();
    // SELECT * FROM user WHERE id_users=1 LIMIT 1
    $userdata = DB::table('user')->where('id_users','=', $user->id)->first();
    $cardata = DB::table('car')->where('id_users','=',$user->id)->first();
} else return redirect( to: '/');
return view( view: 'pages.edit', [
    'user'=>$user,
    'userdata'=>$userdata,
    'cardata'=>$cardata,
]);
```

Рис. 3.46. Вивід даних на сторінку

Для збереження введених даних користувача спочатку необхідно створити масиви, в які будуть передаватись веденні користувачем у відповідні поля значення (рис. 3.47), а потім, перебираємо кожне поле і, якщо користувач ввів у нього нове значення, то зберігаємо його, а якщо користувач змінив вже заповнене поле, то оновлюємо йогозначення в БД (рис. 3.48).

```php
public function edit () {

    if(isset($_POST)&&!empty($_POST)){
        $user = ['name'=>''];
        $userdata = [
            'license'=>'',
            'experiance'=>'',
            'user_photo'=>'',
        ];
        $cardata = [
            'mark'=>'',
            'num'=>'',
            'vin'=>'',
            'class'=>'',
            'full'=>'',
            'trans'=>'',
            'price'=>'',
            'main_photo'=>'',
            'gallery'=>'',
            'coord1'=>'',
            'coord2'=>'',
            'status'=>0,
        ];
```

Рис. 3.47. Створення масивів даних користувача

```php
    foreach($_POST as $k=>$v){
        if(isset($user[$k])&&$v) $user[$k] = $v;
        if(isset($userdata[$k])&&$v) $userdata[$k] = $v;
        if(isset($cardata[$k])&&$v) $cardata[$k] = $v;
    }
    $userid = Auth::user()->id;
    $userdata['id_users'] = $userid;
    $cardata['id_users'] = $userid;
    DB::table('users')->where('id',$userid)->update($user);
    DB::table('user')->updateOrInsert(['id_users'=>$userid],$userdata);
    DB::table('car')->updateOrInsert(['id_users'=>$userid],$cardata);
}
```

Рис. 3.48. Збереження даних користувача

На рис. 3.49 зображений уривок коду в якому відбувається отримання даних з БД про всі додатні користувачами авто, для подальшого відображення

на головній сторінці каталогу. Для цього необхідно перебрати кожне авто в базі даних (рис. 3.50) та вивести необхідну інформацію на сторінці (рис. 3.51).

```php
public function catalog () {

    if (!Auth::check()) return redirect( to: '/');

    $cars = DB::table( table: 'car')->get();

    return view( view: 'pages.catalog', [
        "cars" => $cars,
    ]);
}
```

Рис. 3.49. Отримання даних про авто з БД

```
@foreach($cars as $car)
    @include('inc.net_item', [$car])
@endforeach
```

Рис. 3.50. Перебір кожного авто в БД

```
<a href="/carinfo/{{$car->id}}" class="flax-column main-page-net-item">
    <img src="{{$car->main_photo}}" alt="" class="main-page-net-item-img">
    <p class="small-white-text">
        {{$car->mark}}
    </p>
```

Рис. 3.51. Вивід авто

Результатом зробленої роботи буде відповідність відображення інформації в базі даних (рис. 3.52) та на головному екрані (рис. 3.53).

Марка	Номер авто	Ціна	Власник	Статус
Tesla	AA5975BA	800.00	Dasha	☑
audi q7	AA7777AA	700.00	TEST	☐

Рис. 3.52. Відображення інформації в базі даних

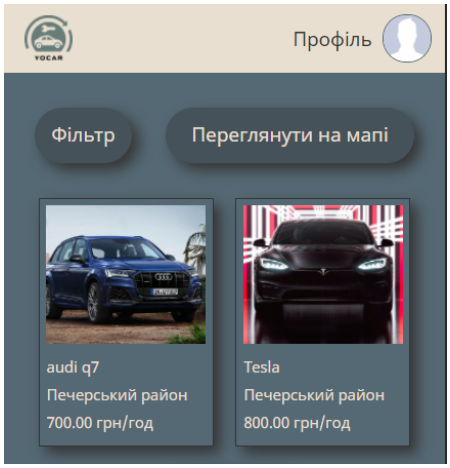

Рис. 3.53. Відображення інформації на головному екрані

3.5. Висновки до третього розділу

У даному розділі було розглянуто етапи створення продукту, обрано та обґрунтовано технології для розробки WEB–додатку, який необхідно реалізувати максимально універсально для кожного користувача. Реалізація фронтенду була досягнута завдяки технологіям CSS та HTML. А завдяки JS було інтегровано у додаток натискання кнопок, введення даних у форму, динамічні стилі та анімація.

Був проведений аналіз технологій розробки серверної частини. На мові PHP та фрейморку Laravel був реалізований весь бекенд. Для релізу мінімально необхідною версією PHP є 7.3. Робота не потребує високих характеристик зі сторони серверу завдяки внутрішнім оптимізаціям фрейморку Laravel. А замість прямих запитів в SQL використовувалося PDO Laravel.

У третьому розділі також було створено унікальний логотип за допомогою графічного редактору Photoshop та повністю розроблено такі сторінки WEB–додатку як:

- сторінка входу;
- сторінка реєстрації;
- сторінка підтвердження реєстрації;
- головна сторінка каталогу авто;

- сторінка фільтрування;
- сторінка мапи;
- сторінка профілю орендодавця;
- сторінка профілю орендаря;
- сторінка редагування профілю;
- сторінка картки авто;
- сторінка перегляду запитів;
- інші спливаючі вікна.

Також було створено БД, налаштовано програмну частину роботи та наведено всі необхідні скріншоти розробки та використання WEB–додатку.

Результатом даного розділу є побудований WEB–додаток з індивідуальним графічним інтерфейсом.

ВИСНОВКИ

Магістерська робота присвячена дослідженню можливих варіантів рішень, спрямованих на вирішення задачі шерінгу власного авто.

В ході виконання роботи були розглянуті вже існуючі на даний момент рішення, які реалізовують можливість орендування авто. На основі цих рішень було складено порівняльну характеристику з урахуванням всіх переваг та недоліків кожного. Було запропоновано варіанти модернізації розроблюваного WEB–додатку, що включають в себе комплексне вирішення основних проблем існуючих продуктів.

Також було проаналізовано предметну область даної роботи та визначено основні вимоги і функції додатку, а також наведений список прецедентів та різних сценаріїв функцій, які можуть зустрітися у WEB–додатку. На основі визначених прецедентів та сценаріїв були побудовані відповідні схематичні рисунки та таблиці.

Зважаючи на вище задані вимоги був проведений аналіз використання існуючих технологій для реалізації WEB–додатку. Відбулося створення структури гіпертекстового документу зі зручним для користувача інтерфейсом та унікальним дизайном, в який інтегровано натискання кнопок, введення даних у форму, динамічні стилі та анімація. Було створено базу даних з таблицями для подальшої реалізації та налаштування програмної частини WEB–додатку. Всі необхідні скріншоти розробки та використання WEB–додатку було наведено та прокоментовано.

Результатом дипломної роботи є побудований WEB–додаток для надання послуг особистого каршерінгу, що містить сторінки: входу, реєстрації, підтвердження реєстрації, каталогу авто, фільтрування, мапи, картки авто, редагування профілю, перегляду запитів, профілю орендодавця та орендаря.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Каршерінг [Електронний ресурс]. Режим доступу: https://sollyplus.rent/blog/chto-takoe-karsherynh/ (дата звернення 02.11.2021) – Назва з екрана.
2. WEB–додаток [Електронний ресурс]. Режим доступу: https://ukr.4meahc.com/what-exactly-is-web-application (дата звернення 03.11.2021) – Назва з екрана.
3. Каршерінг [Електронний ресурс]. Режим доступу: https://carsharingblog.ru/baza/karshering-v-evrope/ (дата звернення 04.11.2021) – Назва з екрана.
4. Getmancar [Електронний ресурс]. Режим доступу: https://getmancar.com.ua (дата звернення 05.11.2021) – Назва з екрана.
5. DriveNow [Електронний ресурс]. Режим доступу: https://www.drivenow.com.au (дата звернення 05.11.2021) – Назва з екрана.
6. Аналіз вимог [Електронний ресурс]. Режим доступу: https://www.drivenow.com.au (дата звернення 14.11.2021) – Назва з екрана.
7. Діаграма прецедентів [Електронний ресурс]. Режим доступу: https://dic.academic.ru/dic.nsf/ruwiki/1299097 (дата звернення 15.11.2021) – Назва з екрана.
8. Багатоплатформність [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Багатоплатформність (дата звернення 26.11.2021) – Назва з екрана.
9. WEB–розробка [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/ru/docs/Learn (дата звернення 27.11.2021) – Назва з екрана.
10. JavaScript [Електронний ресурс]. Режим доступу: https://learn.javascript.ru (дата звернення 27.11.2021) – Назва з екрана.

11. PHP [Електронний ресурс]. Режим доступу: https://www.php.net/manual/ru/intro-whatis.php (дата звернення 27.11.2021) – Назва з екрана.
12. Laravel [Електронний ресурс]. Режим доступу: https://laravel.com/docs/8.x (дата звернення 28.11.2021) – Назва з екрана.
13. Markers Maps JavaScript API [Електронний ресурс]. Режим доступу: https://developers.google.com/maps/documentation/javascript/markers#maps_icon_simple-javascript (дата звернення 29.11.2021) – Назва з екрана.
14. Info Window With maxWidth [Електронний ресурс]. Режим доступу: https://developers.google.com/maps/documentation/javascript/examples/infowindow-simple-max#try-sample (дата звернення 30.11.2021) – Назва з екрана.
15. Style Maps [Електронний ресурс]. Режим доступу: https://snazzymaps.com/style/ (дата звернення 1.12.2021) – Назва з екрана.
16. PDO [Електронний ресурс]. Режим доступу: https://www.php.net/manual/ru/class.pdo.php (дата звернення 4.12.2021) – Назва з екрана.