МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

09 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо- професійної програми «Інформатика»

на тему: «Веб-додаток для автоматизації замовлення авто»

здобувача групи ІН-92 Папіжука Данила Олександровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Данило ПАПІЖУК

(підпис)

Керівник
старший викладач,
кандидат фізико-математичних наук Оксана ШОВКОПЛЯС _____

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-92 Папіжука Данила Олександровича

1. Тема роботи: «Веб-додаток для автоматизації замовлення авто»
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області, формування та визначення мети дослідження. 2) Розгляд технологій, що використовуються для створення веб-додатків. 3) Розроблення веб-додатку для спільного користування автомобілями. 4) Аналіз отриманих результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області, формування та визначення мети дослідження.*		
2	*Розгляд технологій, що використовуються для створення веб-додатків.*		
3	*Розроблення веб-додатку для спільного користування автомобілями.*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 50 стр., 20 рис., 4 табл., 2 додатки, 39 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки вона зосереджена на розробці інформаційних технологій, веб-додатків та сервісів для автоматизації надання спільного користування автомобілями в туризмі.

Об'єкт дослідження – процес визначення бізнес процесів веб-додатків.

Предмет дослідження – методи розробки веб-додатків для автоматизації оренди автомобілів.

Мета роботи – розробка веб-додатку для автоматизації оренди автомобілів в сфері туризму.

Методи дослідження – використання інформаційно-комунікаційних технологій для проектування та побудови веб-додатку, розробка програмного забезпечення та теорія проектування бази даних.

Результати – розроблено веб додаток для автоматизації оренди автомобілів у сфері туризму, який дозволяє обирати автомобіль на певний період часу, управляти списком наявних послуг і обмежувати дозвіл користувачам за потреби.

ВЕБ ДОДАТОК, ЕЛЕКТРОНА КОМЕРЦІЯ, JAVA, SPRING, ОРЕНДА АВТОМОБІЛІВ, АВТОМАТИЗАЦІЯ

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки визначає зростаючий попит на зручні та ефективні способи оренди автомобілів у сучасному світі. Використання веб-додатку для автоматизації цього процесу має потенціал поліпшити доступність, ефективність та зручність для клієнтів і компаній, що займаються орендою автомобілів.

Об’єкт дослідження. Процес визначення бізнес процесів веб-додатків.

Предмет дослідження. Методологія розробки зручних веб-додатків для поліпшення оренди автомобілів.

Гіпотеза. Автоматизації оренди автомобілів можна досягнути шляхом використання інформаційної технології, що реалізує веб-додаток для розміщення необхідних послуг з оренди автомобілів.

Наукова новизна. Описане у даній роботі програмне рішення дозволить досягти більшої ефективності в побудові інформаційної системи компанії, дозволяючи автоматизувати внутрішні процеси та спростити аналіз та візуалізації статистичних даних.

Апробація матеріалів роботи. Основні результати роботи оприлюднені та обговорені на міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, математика, автоматика» (IMA – 2023).

Структура. Дана робота складається зі вступу, аналітичного огляду, вибір методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

Зв'язок роботи з науковою темою. Кваліфікаційна робота виконана на кафедрі комп’ютерних наук та пов’язана з виконанням науково-дослідної роботи № 0118U006971 «Методи, математичні моделі та інформаційні технології аналізу і синтезу інфокомунікаційних систем» (2018-2023).

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасний стан

Інформаційно-комунікаційні технології (ІКТ) – це сукупність технологій (рис. 1.1), які дозволяють отримувати, обробляти та обмінюватися інформацією [1, 2]. ІКТ мають велике стратегічне значення в економічній, соціальній, культурній та інших сферах.

Рисунок 1.1 – Компоненти ІКТ [1, 2]

Електронна комерція (e-commerce) використовує інформаційні та комунікаційні технології для збору, обробки та обміну інформацією з метою онлайн-торгівлі та ведення бізнесу через Інтернет [3, 4]. У 1992 році був запущений перший сайт електронної комерції під назвою Book Stacks Unlimited (рис. 1.2) [5, 6]. За цим успіхом послідував запуск eBay та Amazon у 1995 році. Незважаючи на плин часу, електронна комерція існує і залишається актуальною і сьогодні. Однак не кожен бізнес або постачальник послуг приймає цю форму торгівлі через виклики, пов'язані з впровадженням цієї інновації. Пошук надійного логістичного партнера, розрахунок податків та створення зручного комерційного додатку – це лише деякі з викликів. Тим не менш, вона стала ключовим явищем у сучасному світі. Електронна комерція дозволяє здійснювати

торгівлю без обмежень у просторі та часі, створюючи глобальний ринок, доступний мільйонам людей по всьому світу.

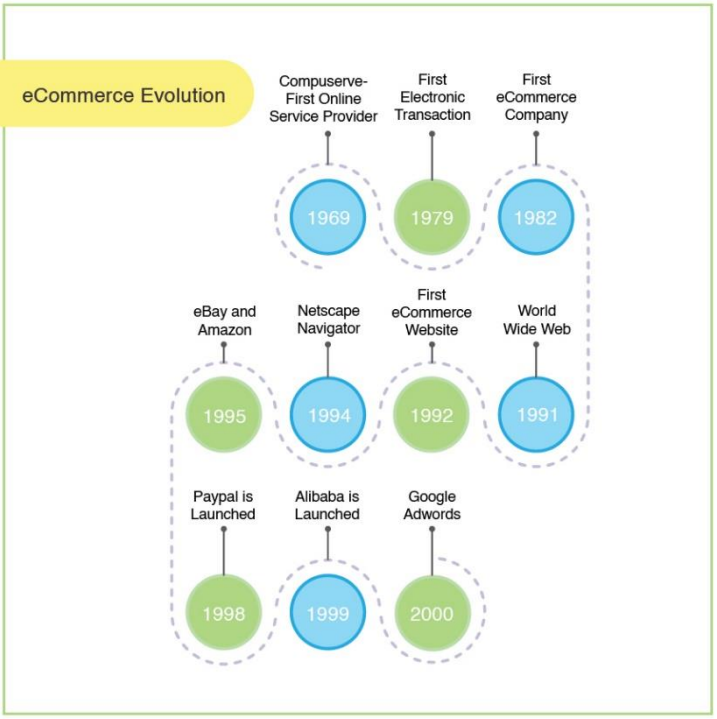

Рисунок 1.2 – Еволюція електронної комерції [5, 6]

Розвиток електронної комерції тісно пов'язаний із впливом Індустрії 4.0 – нової ери технологічних інновацій та цифрової трансформації [7–9]. Індустрія 4.0 характеризується широким використанням автоматизації, Інтернету речей (IoT), штучного інтелекту (ШІ), аналітики даних, хмарних технологій та інших передових технологій. Загалом, Індустрія 4.0 пропонує нові можливості для електронної комерції з точки зору автоматизації та оптимізації бізнес-процесів. Впровадження роботів та автоматизованих систем у складському господарстві та логістиці може підвищити швидкість і точність обробки замовлень та зменшити витрати на робочу силу.

ІКТ та електронна комерція змінюють традиційні бізнес-моделі та способи ведення бізнесу. Це створює нові можливості для підприємців, дозволяючи їм залучати клієнтів з усього світу, знижувати маркетингові та ресурсні витрати, підвищувати операційну ефективність, сприяти швидкому розвитку бізнесу та

створювати інноваційні продукти і послуги. Електронна комерція також змінює спосіб нашого споживання, пропонуючи споживачам широкий вибір товарів і послуг, зручність здійснення покупок і можливість порівняти ціни та характеристики товарів перед тим, як зробити покупку.

Загалом, електронна комерція є ключовим фактором у сучасному економічному та соціальному ландшафті. Вона створює нові можливості для бізнесу, підтримує економічне зростання та сприяє глобальному зв'язку та співпраці. Тому розуміння та використання інформаційно-комунікаційних технологій в контексті електронної комерції стає важливим фактором успіху для бізнесу та споживачів у цифрову епоху.

Оренда автомобілів для подорожей є однією з послуг, що пропонуються через електронну комерцію (рис. 1.3) [10, 11]. Цей динамічний сегмент електронної комерції набуває все більшої популярності завдяки своїм перевагам та зручності для клієнтів [12].

Рисунок 1.3 – Процес зв’язку е-комерції та клієнту

Інтернет-торгівля, яка є одним з ключових напрямків електронної комерції, дозволяє компаніям представляти свої автомобільні послуги через різні електронні платформи. До них відносяться інтернет-магазини, де клієнти можуть

переглядати широкий асортимент автомобілів і робити замовлення в зручний для них час. Набувають популярності також електронні аукціони та маркетплейси, де замовлення на автомобілі можна зробити швидко та ефективно.

Такий формат роздрібної торгівлі сприяє залученню нових клієнтів, оскільки вони мають можливість ознайомитися з різними пропозиціями та зробити свій вибір, не обмежуючись територіальним розташуванням [13]. Крім того, робота компаній у режимі 24/7 забезпечує постійну доступність автомобільних послуг, що є надзвичайно зручним для клієнтів з різних часових поясів та графіків [14–16].

Онлайн-торгівля пропонує споживачам велику зручність, дозволяючи їм замовляти автомобілі з будь-якого місця за допомогою пристроїв з доступом до Інтернету, таких як комп'ютери, смартфони та планшети [17]. Це дає можливість зручно і швидко забронювати автомобіль для відпустки або ділової поїздки, без необхідності відвідувати фізичний офіс або зв'язуватися з операторами по телефону.

Крім того, електронна комерція охоплює й інші сфери, які допомагають розширити можливості бізнесу та задовольнити потреби споживачів. Наприклад, електронний банкінг та електронні платежі забезпечують зручні та безпечні фінансові операції в Інтернеті. Цифровий маркетинг та онлайн-реклама дозволяють компаніям привертати увагу клієнтів за допомогою рекламних кампаній та персоналізованих пропозицій [18].

Розвиток електронної комерції спонукає компанії впроваджувати інноваційні рішення та технології. Наприклад, штучний інтелект використовується для вдосконалення систем персоналізації та рекомендацій, щоб допомогти клієнтам знаходити більш точні та релевантні товари та послуги. Аналітика даних дозволяє компаніям отримувати цінну інформацію про споживачів та їхню поведінку, що сприяє ефективному управлінню запасами та прогнозуванню попиту [14, 19].

Використання передових технологій, таких як хмарні сховища, криптографічні методи та технології аутентифікації, має велике значення для електронної комерції [15]. Ці технології забезпечують масштабованість, безпеку та довіру до віртуальних транзакцій, дозволяючи зберігати та обробляти великі обсяги інформації без значних інфраструктурних витрат. Конфіденційність і безпека транзакцій є важливими для бізнесу, забезпечуючи надійність електронних платежів і запобігаючи кібершахрайству.

1.2 Аналіз принципів побудови інтернет-магазинів

Створення інтернет-магазину вимагає системного та поетапного підходу (рис. 1.4) [20, 21]. Кожен етап має свої важливі обов'язки та впливає на кінцевий результат. Ретельна увага до кожного кроку допоможе забезпечити ефективну та успішну реалізацію інтернет-магазину.

Рисунок 1.4 – Етапи побудови інтернет-магазину

Перейдемо до детального розгляду етапів побудови інтернет-магазину:

- **Обрати платформу для електронної комерції.** Вибір платформи електронної комерції є невід'ємною частиною ведення успішного бізнесу. Платформа виступає в ролі центру управління, надаючи необхідні інструменти для досягнення оптимальних результатів онлайн-продажів і забезпечення задоволеності клієнтів. Важливими характеристиками якісної платформи є її здатність забезпечити високий рівень юзабіліті, ефективну підтримку клієнтів, спрощений процес замовлення та стабільний веб-хостинг. При виборі програмного забезпечення слід враховувати як майбутні, так і поточні потреби бізнесу для забезпечення успішної роботи.
- **Обрати цільову аудиторію.** Ідентифікація цільової аудиторії є

важливим кроком у визначенні групи людей, зацікавлених у товарах чи послугах, що пропонуються. При виборі цільової аудиторії слід керуватися трьома основними категоріями: демографічними характеристиками (такими як вік, стать, дохід), географічним розташуванням (з урахуванням можливості доставки і надання послуг) та інтересами аудиторії (створення психологічного портрета клієнта). Прогнозованою цільовою аудиторією можуть бути активні мандрівники віком від 18 до 45 років.

- **Обрати продукцію.** Один з найважливіших і найскладніших етапів, оскільки більшість підприємців стикаються з проблемою визначення товарів або послуг, які вони повинні пропонувати на ринку. Продукція повинна бути не тільки рентабельною, але й приносити прибуток підприємцю як основну мету його бізнесу. Щоб уникнути проблем на цьому етапі, доцільно ознайомитися з сучасними тенденціями на ринку онлайн-продажів і визначити тип продукту, який буде привабливим для підприємця. У нашому конкретному випадку таким продуктом є оренда автомобілів для подорожей.
- **Розробити сайт.** Для того, щоб створити привабливий веб-додаток [22], який слугуватиме зручним середовищем для клієнтів для вибору необхідних їм продуктів, слід виконати кілька ключових кроків. Перш за все, необхідно сформулювати бізнес-ідею, яка відповідає концепції компанії. Далі слід обрати відповідну назву бізнесу, яка відображає його суть та привертає увагу [18, 19]. Важливим аспектом є створення логотипу, який привертає увагу та виділяється. Крім того, веб-додаток повинен бути багатим на контент, де текстові елементи доповнюються ефектними ілюстраціями, що стимулюють інтерес клієнтів. Привернення уваги зацікавлених сторін через вдале поєднання цих етапів допоможе створити ефективний і привабливий веб-додаток [19].
- **Підключити платіжний шлюз.** Під час підключення платіжного

шлюзу для забезпечення ефективної роботи електронної комерції слід врахувати кілька аспектів. По-перше, вам слід ретельно визначитися з варіантами оплати, вибравши постачальника платіжних послуг, через якого будуть оброблятися транзакції. Вам також потрібно буде вирішити, в якій валюті будуть надаватися товари або послуги. Також варто визначити вартість комісій, пов'язаних з використанням конкретного постачальника платіжних послуг.

– **Обрати структуру бізнесу.** Вибір оптимальної структури для організації бізнесу є важливим аспектом для підприємця, оскільки забезпечення легальності бізнесу є важливим фактором. У зв'язку з цим вибір структури має вирішальне значення. Найпоширенішими варіантами є одноосібне володіння, партнерство та товариство з обмеженою відповідальністю. Враховуючи особливості малого бізнесу, рекомендується обирати приватну організаційно-правову форму.

Після того, як ви виконали всі кроки, можна починати підготовку магазину до запуску. Одним з необхідних кроків є надання магазину домену, на якому він буде працювати (наприклад, .ua, .com тощо). Останнім кроком є перевірка коректного функціонування магазину, що включає такі аспекти, як відображення товарів на сторінках сайту та проведення платежів. Продуктивність можна оцінити за допомогою інструменту Google PageSpeed Insights (табл. 1.1) [23].

Таблиця 1.1 – Метрики PageSpeed Insights

Назва	**Опис**
1	**2**
First Contentful Paint (FCP)	Вимірює час, який потрібно для відображення першого елемента контенту на екрані. Вказує, наскільки швидко користувачі бачать перший зміст на сторінці.

Продовження таблиці 1.1

1	2
Largest Contentful Paint (LCP)	Вимірює час, який потрібно для відображення найбільшого елемента контенту на екрані, такого як зображення або блок тексту. Вказує на загальний час завантаження сторінки.
Cumulative Layout Shift (CLS)	Вимірює ступінь візуальної стабільності сторінки шляхом визначення сумарних зміщень елементів під час завантаження. Вказує, наскільки часто елементи сторінки зміщуються, що може призвести до неприємного користувацького досвіду.
Time to First Byte (TTFB)	Вимірює час, який потрібно серверу для надсилання першого байта відповіді після отримання запиту. Вказує на час, який потрібний для встановлення з'єднання між клієнтом і сервером.
Total Blocking Time (TBT)	Вимірює загальний час блокування сторінки, коли виконання JavaScript-коду блокує взаємодію користувача зі сторінкою. Вказує на час, протягом якого сторінка стає нереагуючою на дії користувача.
Speed Index	Вимірює, як швидко контент сторінки візуально відображається на екрані. Вказує на загальну швидкість завантаження сторінки з огляду на видимість змісту.

1.3 Аналіз аудиторії

Перш ніж приступити до розробки майбутнього веб-додатку для автоматизації прокату автомобілів під час подорожей, необхідно звернутися до громадської думки. У зв'язку з цим було проведено опитування з метою вивчення поточної ситуації в туристичній галузі. Опитування проводилося серед представників обох статей, чоловіків та жінок, які представлені майже порівну (по 49,2% кожної статі) (рис. 1.5). В опитуванні також взяли участь деякі респонденти (1,5%), які відмовилися ідентифікувати себе з будь-якою гендерною групою.

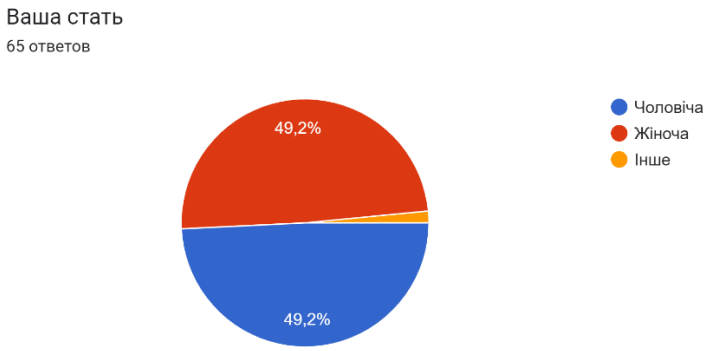

Рисунок 1.5 – Стать респондентів

Більшість респондентів були дорослими: 75,4% – у віковій групі 18-20 років, 10,8% – у віковій групі 21-25 років, а решта – молодші 18 або старші 25 років (рис. 1.6).

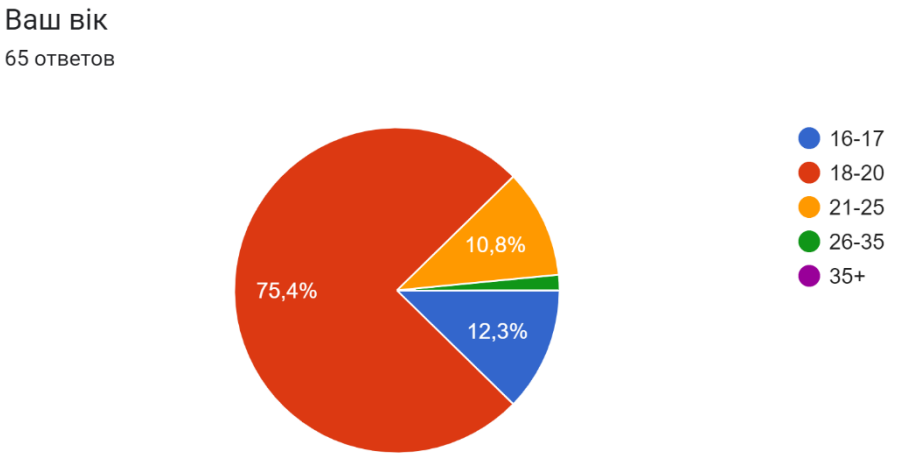

Рисунок 1.6 – Віковий діапазон

Питання про частоту подорожей (рис. 1.7) отримало середню відповідь, що свідчить про те, що люди загалом цікавляться галуззю. Крім того, ми запитали, чи користуються респонденти додатками, і якщо так, то якими саме. Більшість респондентів згадали такі додатки, як Google Maps, Booking та Укрзалізниця. Також були поодинокі випадки використання TripAdvisor та Moovit.

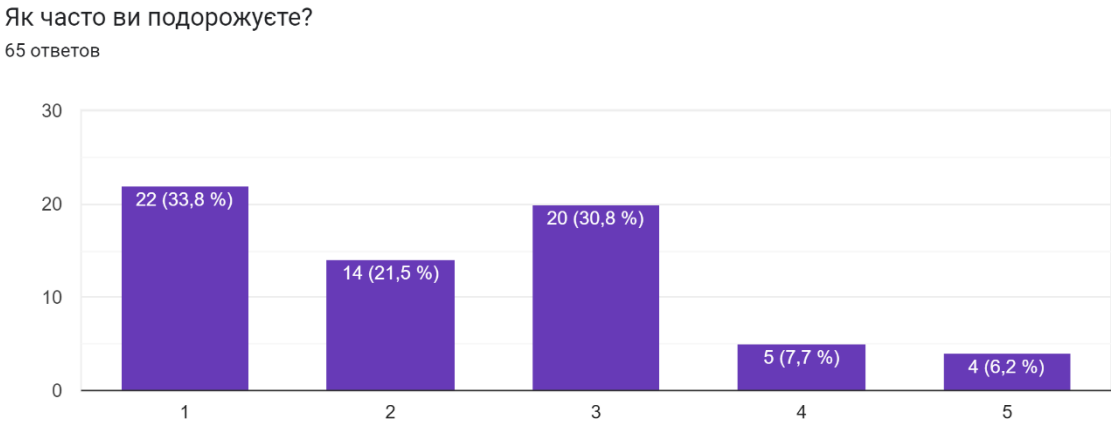

Рисунок 1.7 – Частота подорожей (1 – Дуже рідко, 5 – Дуже часто)

Серед ключових критеріїв (рис. 1.8), які найчастіше згадуються, актуальність інформації, ціна, простота використання та швидкість.

Рисунок 1.8 – Критерії додатків

На запитання про використання автомобіля під час подорожі 66,2% респондентів відповіли "так", 20% – "можливо" і 12,3% – "ні" (рис. 1.9). Це свідчить про те, що існує певне підґрунтя для позитивної думки про оренду автомобіля під час подорожей. Респонденти наводили цитати на підтвердження своєї думки. Деякі з них були позитивними: "Автомобіль – найзручніший засіб пересування", "Я завжди подорожую автомобілем", "Це зручний спосіб пересування, який можна легко підлаштувати під свій графік і побажання", "Власний автомобіль прискорює знайомство з великим містом". Проте були й

негативні відповіді: "Поїздка потягом – вигідніший і комфортніший варіант, особливо для тих, хто не має власного авто, оскільки громадський транспорт має регулярний графік і забезпечує більш зручний і швидкий переїзд", "Поїздка автомобілем коштує дорожче і є більш небезпечною, ніж потягом", "Забруднення повітря плюс високі витрати".

Рисунок 1.9 – Зацікавленість використання автомобіля

Останнє питання запитувало думку щодо надання власного автомобіля в користування іншим особам, подібно до BlaBlaCar (рис. 1.10). Більшість відповіли "ні" на це питання – 60,3%, а 27% не мали чіткої думки. Це свідчить про те, що користувачам важко зробити свої автомобілі доступними для масового використання.

Рисунок 1.10 – Надання власного автомобіля у користування

1.4 Інструменти для розробки веб додатків

У сфері розробки веб-додатків існує безліч готових конструкторів, якими ви можете скористатися. Вони відрізняються за вартістю, надійністю та доступністю. Серед безлічі таких конструкторів можна виділити два особливо зручних і надійних варіанти: Wix та Shopify (рис. 1.11) [20, 21]. Перший підходить для малого бізнесу, а другий – для великих підприємств. Однак обидва конструктори збільшують вартість користування своїми послугами, тому варто розглянути дешевші або навіть безкоштовні альтернативи.

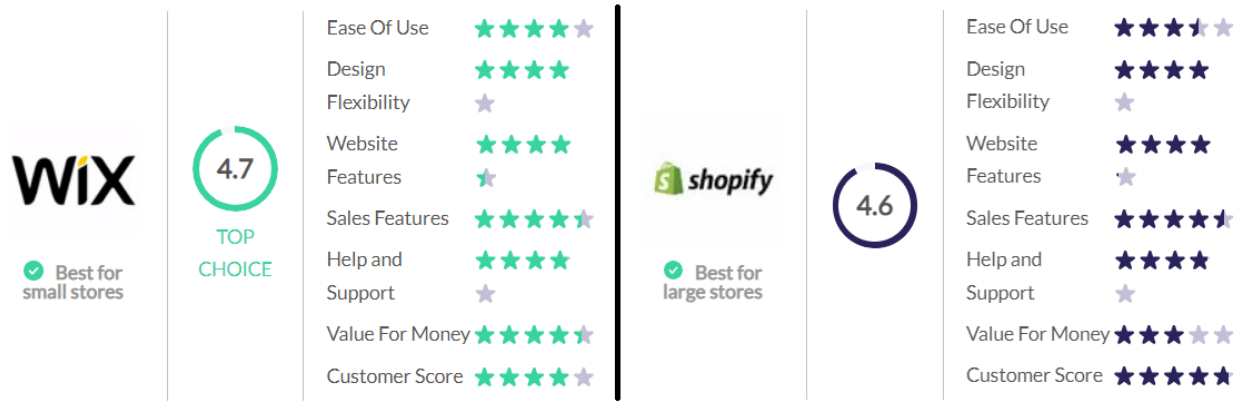

Рисунок 1.11 – Порівняння готових конструкторів веб-додатків [20, 21]

Для розробників, у тому числі студентів, доступ до різноманітних безкоштовних інструментів розробки веб-додатків надається широким спектром ресурсів в Інтернеті. Цей інструментарій включає такі популярні та визнані інструменти, як REST-assured, Notepad++, XAMPP, Visual Studio Code, Figma, Postman, Angular, Django, Bootstrap, GitHub, Sublime Text та багато інших [24]. Ці програми пропонують різноманітні функціональні можливості, включаючи можливість запускати локальні сервери, створювати дизайн для веб-сторінок і послідовних продуктів, а також можливість розробляти основу веб-сайту. Слід також зазначити, що на додаток до цих інструментів, деякі мови програмування надають вбудовану функціональність для створення і розгортання веб-додатків.

1.5 Постановка задачі

Метою роботи є розробка веб-додатку для автоматизації оренди автомобілів в сфері туризму

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Провести огляд інструментів розробки та вибрати найбільш оптимальні з них;
2) розробити базу даних для зберігання інформації, необхідної для роботи веб-додатка;
3) розробити інтерфейс веб-додатка, що буде зручним для користувача;
4) здійснити проектування архітектури веб-додатка з метою реалізації надання послуг;
5) розробити алгоритм роботи веб-додатка;
6) виконати тестування веб-додатка з метою перевірки його функціональності та надійності.

При успішному виконанні поставлених завдань буде забезпечена можливість відстежування наявних послуг, замовлення та надання їх користувачам.

2 ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Процес розробки

Перед тим, як проектувати майбутній проект, необхідно обрати підхід до розробки програмного продукту. В даному випадку для цього буде використана модель Scrum. Існує ряд аспектів (табл. 2.1), які сприяють перевазі Scrum як кращого вибору. У ролі Scrum Master та Product Owner буде залучено дипломного керівника, який виконає завдання формулювання чітких вимог та полегшить процес залучення до розробки веб-додатку.

Таблиця 2.1 – Аспекти Scrum [25, 26]

Назва	Опис
Ітеративний підхід	Розбиття проекту на короткі ітерації (спринти) з чітко визначеними цілями. Це дозволяє швидко отримувати функціонал, прототипи та зворотний зв'язок від користувачів
Гнучкість та адаптивність	Гнучкість у виборі та встановлені пріоритету функціоналу. Завдяки регулярним зустрічам (Scrum-планування, щоденні зустрічі, огляди спринту) можна швидко адаптуватись до змін вимог, виявити проблеми та забезпечити високу якість продукту.
Командна співпраця	Передбачає активну комунікацію та співпрацю між різними ролями, такими як продуктовий власник, Scrum-мастер та команда розробників. Сприяє кращому розумінню вимог, ефективному управлінню ризиками та більшому залученню всіх зацікавлених сторін.
Часті випуски	Завдяки ітераційному підходу Scrum, можна швидко впроваджувати новий функціонал та випускати оновлення продукту на регулярній основі. Це дозволить швидко реагувати на змінні потреби ринку та отримувати важливий зворотний зв'язок від користувачів.
Складні завдання	Структурований підхід до управління складними завданнями, розбиваючи їх на менші, керовані спринтами. Це допомагає зменшити ризик та зберегти контроль над процесом розробки. Розробники можуть фокусуватись на виконанні окремих завдань та досягненні проміжних цілей на кожному спринті.
Задоволення клієнта	Ставить акцент на взаємодію з продуктовим власником та залучення замовників на регулярні огляди спринтів. Це дозволяє швидко отримувати фідбек від замовників, змінювати пріоритети та забезпечувати високу задоволеність їх потреб.
Покращення процесу	Систематичні огляди спринтів та ретроспективи для постійного вдосконалення процесу розробки. Це дозволяє розробникам ідентифікувати проблеми, виявляти можливості для вдосконалення та впроваджувати зміни для підвищення продуктивності та якості.

Scrum зазвичай вважається потужною гнучкою методологією, яка довела свою ефективність у контексті розробки веб-додатків, таких як системи оренди автомобілів. Цей підхід забезпечує організаційну структуру, гнучкість і акцент на активній взаємодії з клієнтом, що сприяє успішному управлінню проектом і досягненню поставлених цілей.

2.2 Проектування бази даних

База даних є невід'ємною частиною будь-якого проекту, в якій зберігаються не лише дані про клієнтів, але й дані про продукцію та інші важливі аспекти. Перш ніж почати, потрібно вибрати систему управління базами даних (СУБД). Сьогодні існує багато СУБД, зокрема MySQL, PostgreSQL, Oracle, MariaDB, Microsoft SQL Server, MongoDB, Redis, SQLite, IBM DB2 та Elasticsearch [27]. Вони відрізняються за сферою застосування. В даному випадку ми використовуємо MySQL [28], яка є безкоштовною, з відкритим вихідним кодом і має велику активну спільноту, готову відповісти на будь-які питання, що можуть виникнути під час роботи з нею.

Перед початком роботи необхідно розробити концептуальну модель, яка відображає загальний принцип роботи веб-додатку (рис 2.1).

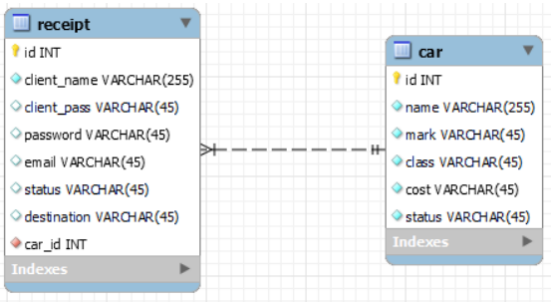

Рисунок 2.1 – Концептуальна модель бази даних веб-додатку

Після того, як концептуальна модель завершена, необхідно створити логічну та фізичну моделі (рис. 2.2). Цей процес включає нормалізацію бази даних для досягнення третьої нормальної форми. Загалом існує шість нормальних форм (далі – НФ) [29], які використовуються для опису гнучкості

використання бази даних. Розглядаючи існуючу концепцію, стає очевидним, що деякі сутності мають надлишкові атрибути. Тому в цьому випадку виконується нормалізація для отримання третьої нормальної форми.

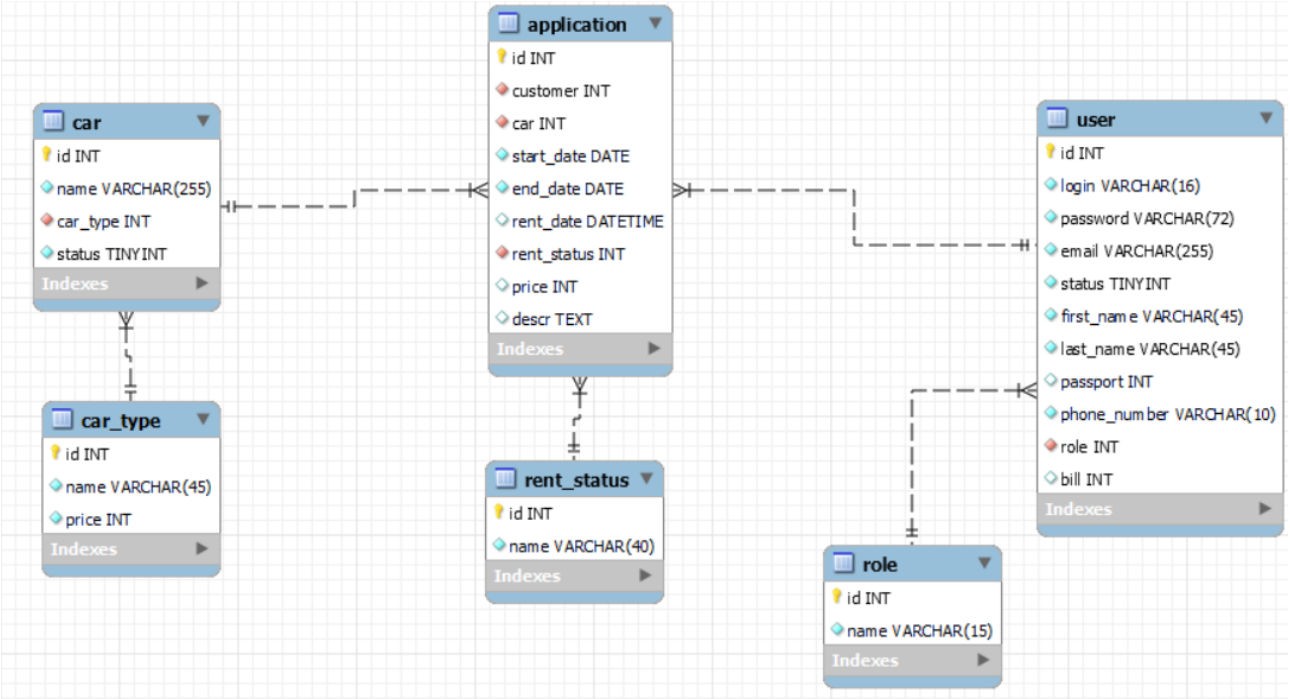

Рисунок 2.2 – Нормалізована модель бази даних веб-додатку

2.3 Огляд інструментів для розробки

Для розробки програмного забезпечення на мові програмування Java вам знадобляться певні інструменти (табл. 2.2).

Вибір цих інструментів ґрунтується на їхньому зручному та інтуїтивно зрозумілому інтерфейсі. Крім цих додатків, при написанні коду використовуються специфічні патерни проектування – типові методи вирішення конкретних завдань, а також API – набір програмного коду для передачі даних між програмними продуктами [30, 31] (рис. 2.3).

Таблиця 2.2 – Інструменти для розробки

Інструмент	Опис
IntelliJ IDEA [32]	Інтегроване середовище розробки забезпечує зручну можливість написання коду на мові Java, надаючи при цьому розширені функціональність та зручні надбудови [33].
WebStorm [34]	Пропонується створення інтегрованого веб-середовища розробки, яке забезпечує зручну можливість моніторингу коду, написаного для сторінок веб-додатку [33].
Apache Tomcat [35]	Майбутня серверна інфраструктура, на якій розгорнуто веб-додаток.
MySQL Workbench [28]	Зручний засіб для взаємодії з системою керування базами даних MySQL.
Figma [36]	Безкоштовний онлайн сервіс, який дозволяє створювати дизайн майбутнього веб-додатку.

До переліку майбутніх патернів входять Singleton, Command, Iterator, Builder, Abstract Factory. Серед майбутніх API – JDBC (Java Database Connectivity) [37], який дозволяє мові програмування Java взаємодіяти з базою даних і представлений як реалізація фреймворку Hibernate ORM. На додаток до вищезазначеного, фреймворк Spring відіграватиме важливу роль у забезпеченні майбутнього додатку портативною функціональністю, безпекою та високою продуктивністю [38].

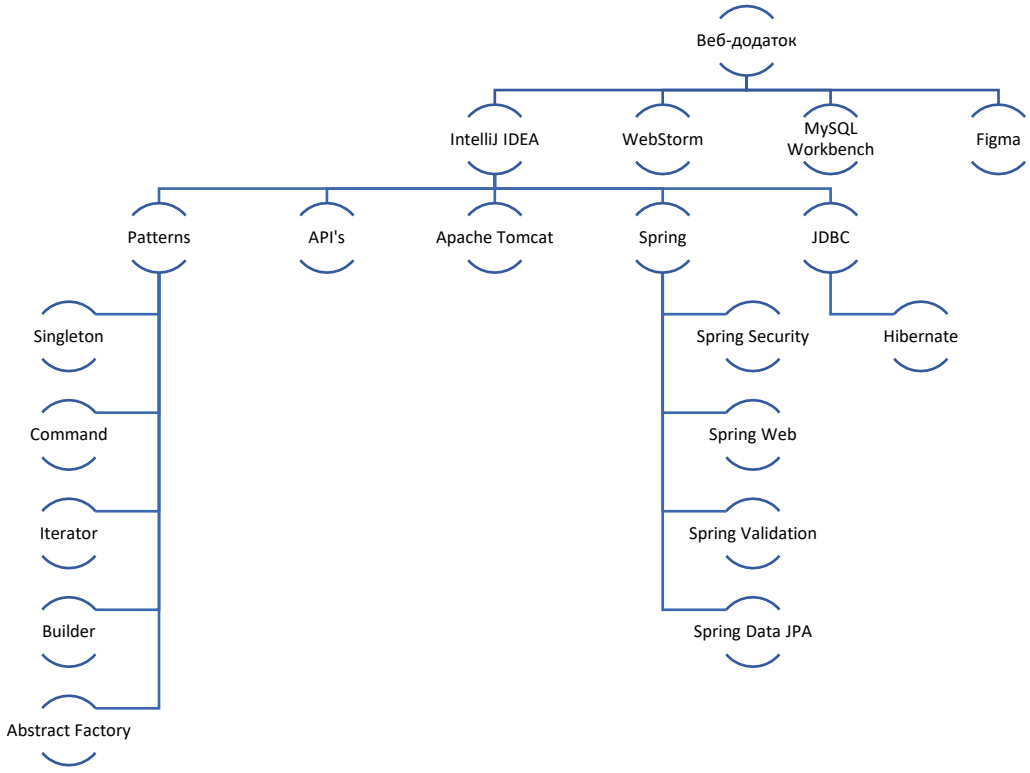

Рисунок 2.3– Схема використання інструментів для розробки

2.4 Клієнтська частина веб-додатку

Одним з атрибутів якісного веб-додатку, який доповнює його функціональність, є зрозумілий та естетичний дизайн. Розроблюваний додаток буде характеризуватися простим і швидким дизайном, що полегшить користувачеві доступ до всіх необхідних функцій і допоможе йому зрозуміти його призначення. Прикладом такого дизайну є веб-сайт BlaBlaCar (рис. 2.4) [39].

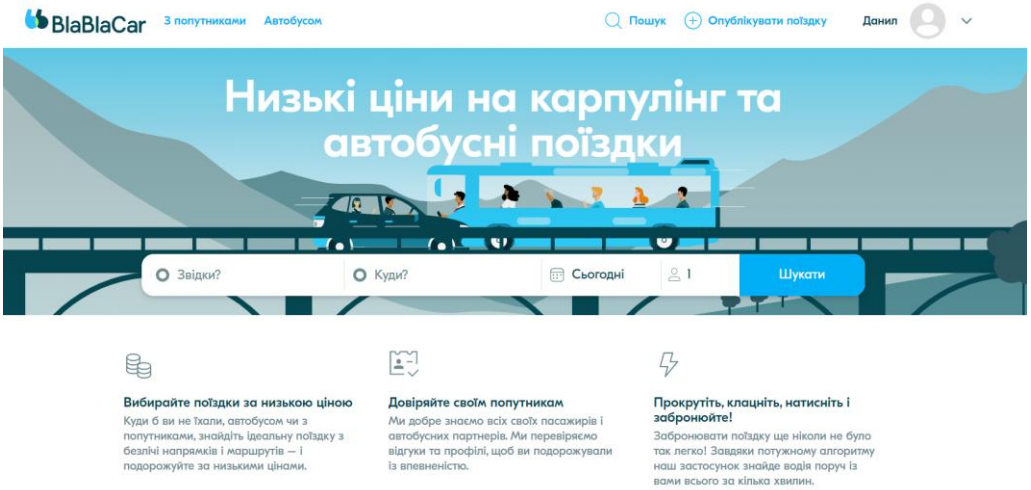

Рисунок 2.4 – Приклад дизайну

При розробці майбутнього веб-додатку будуть уважно проаналізовані та враховані актуальні тенденції у галузі дизайну, що присутні у сучасному світі (рис. 2.5).

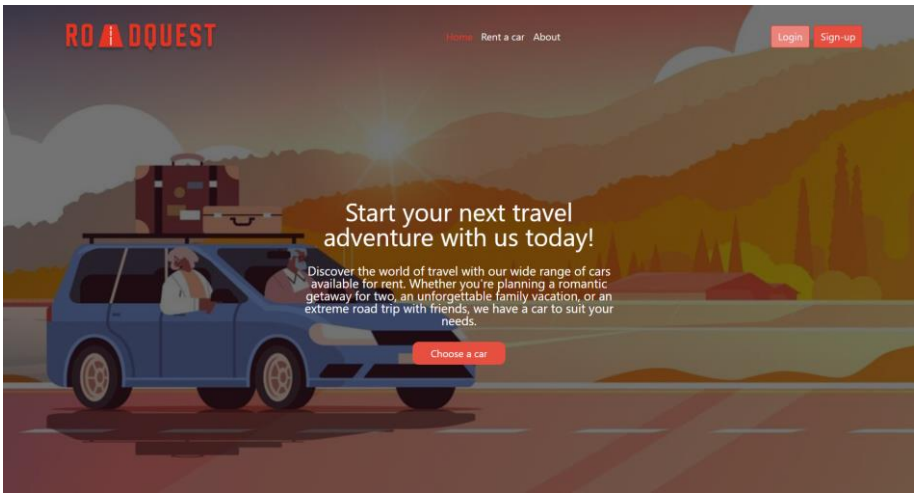

Рисунок 2.5 – Приклад майбутнього дизайну

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Налаштування бібліотек

Для досягнення надійності та стабільності роботи системи вимагається встановити необхідні програмні бібліотеки. Серед найважливіших бібліотек, які використані в реалізації, можна виділити Lombok, MySQL Connector,Spring Security, Spring Validation, Spring Web, Spring Data JPA та Spring Thymeleaf. Ці бібліотеки надають додаткові функції та можливості для реалізації безпеки, валідації даних, веб-розробки, роботи з базою даних та шаблонізації відповідно. Для забезпечення правильного підключення цих бібліотек використовується інструмент Maven, який здатний автоматизувати процес управління залежностями через конфігураційний файл pom.xml (ДОДАТОК Б).

3.2 Архітектура додатку

Добре продумана архітектура папок впливає на ефективність, модульність, розширюваність і ремонтопридатність програмного продукту (рис 3.1, табл. 3.1). Вона забезпечує чітку та логічну організацію файлів і компонентів системи, полегшує співпрацю між розробниками, а також забезпечує зручну навігацію та управління проектом. Правильна структура каталогів також допомагає забезпечити чистоту коду, дотримання принципів розробки та стандартів програмування, що впливає на якість і підтримку проекту протягом усього його життєвого циклу.

Таблиця 3.1 – Характеристика пакетів проекту

Назва пакету	**Опис**
1	**2**
Controller	Зберігає класи для відстеження кінцевих точок url адрес, які повертають результат в залежності від точки.
Entity	Зберігає класи-сутності 1:1 до сутностей із бази даних.
Model	Зберігає класи моделі для обробки під час створення нової сутності у базі даних.
Repository	Методи запитів до бази даних.

Продовження таблиці 3.1

1	2
Security	Механізм захисту проекту та розмежування доступу на основі ролей користувачів (адміністратор, менеджер, клієнт).
Service	З'єднуюча ланка між базою даних та контролерами, яка оброблює помилки та повертає відформатований результат запиту з бази даних.
Util	Зберігає додаткові інструменти для забезпечення зручного написання програмного коду. Наприклад, константи значень відповідних сторінок.
Static	Зберігає додаткові інструменти для стабільної роботи проекту, такі як стилі сторінок, скрипти, зображення на сторінках та скрипт створення та заповення бази даних значеннями.
Templates	Зберігає сторінки відображення проекту (сторінки користувача, менеджера та адміністратора)

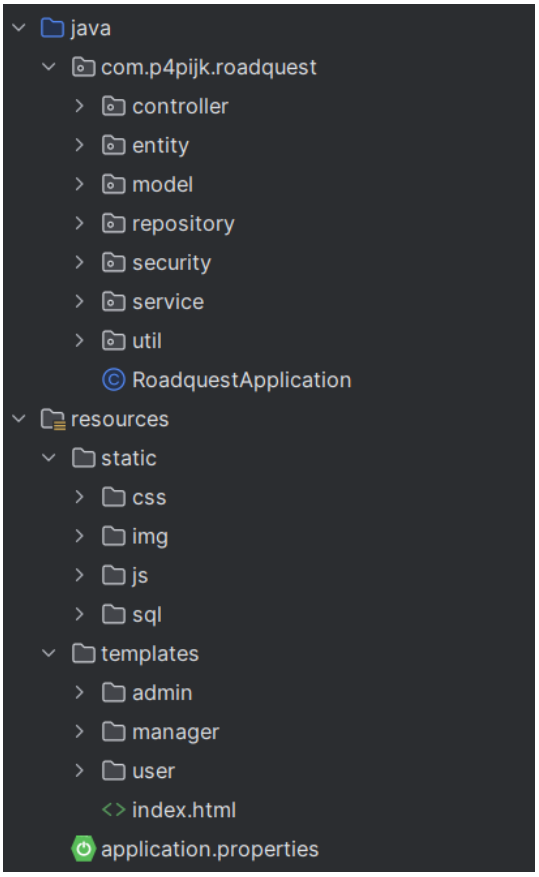

Рисунок 3.1 – Архітектура пакетів

3.3 Підключення до бази даних

Щоб надати доступ до бази даних, ви повинні ввести відповідні конфігураційні дані у файл application.properties (рис. 3.2). Цей файл є важливою частиною програмної реалізації, що зберігає параметри, які дозволяють підключитися до бази даних і налаштувати відповідні параметри доступу. Розміщення відповідної інформації у файлі application.properties забезпечує зручний і централізований спосіб управління налаштуваннями бази даних в контексті програмного додатку.

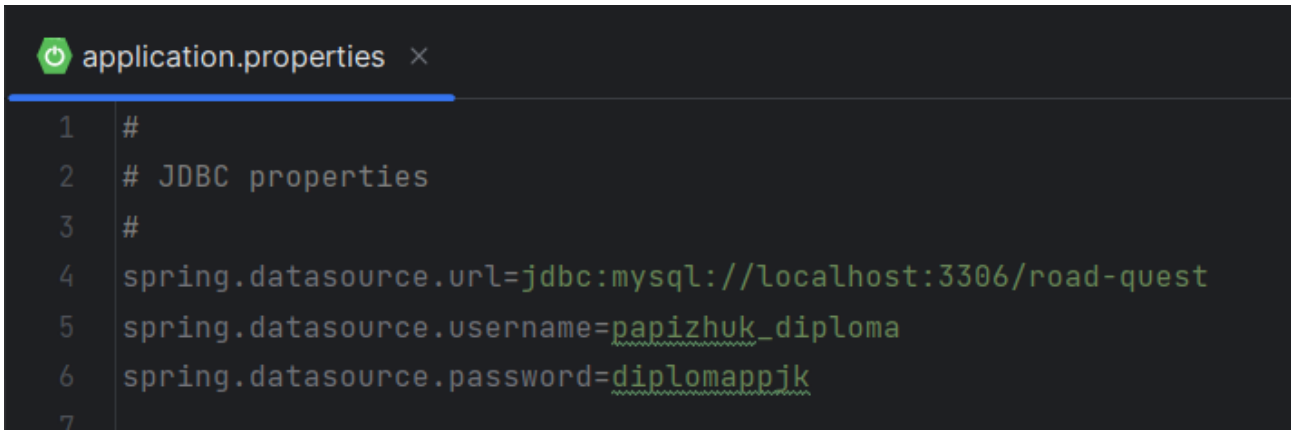

Рисунок 3.2 – З’єднання з базою даних

3.4 Механізм захисту

В рамках конфігурації механізмів безпеки створюється спеціальний клас (ДОДАТОК А – UserSecurityConfig.java), який відповідає за управління діями користувача та визначення доступу до різних сторінок додатку в залежності від призначеної користувачеві ролі. Цей клас визначає права доступу, які дозволяють або обмежують взаємодію з певними функціональними елементами системи.

3.5 Авторизація та реєстрація

У контексті розробки веб-додатку для оренди автомобілів важливим етапом є реалізація механізму авторизації та реєстрації користувачів. Під час реєстрації пароль шифрується та зберігається в базі даних за допомогою алгоритму bcrypt

(ДОДАТОК А – UserSecurityConfig.java, LoginController.java, SignUpController.java). Під час авторизації механізм безпеки розшифровує та звіряє введений користувачем пароль зі збереженим значенням. Це забезпечує безпеку та конфіденційність даних користувача під час користування додатком.

3.6 Панель адміністратора

Розробка адмін-панелі є важливим етапом у процесі розробки веб-додатку для прокату автомобілів. Ця панель виступає стартовим і основним компонентом системи, надаючи широкий спектр функціональних можливостей і забезпечуючи повне управління і контроль над усіма аспектами замовлення оренди автомобіля.

Панель адміністратора (ДОДАТОК А – AdminController.java) містить різноманітні налаштування та інструменти, які дозволяють адміністратору ефективно керувати процесами прокату автомобілів. Це включає в себе управління та моніторинг автомобілів, реєстрацію нових менеджерів та надання користувачам доступу до сервісу.

Крім того, панель адміністратора виступає батьківським елементом для інших панелей, включаючи панель менеджера і панель користувача. Це означає, що багато налаштувань і функцій, доступних в панелі адміністратора, можуть бути успадковані і використані в інших панелях з мінімальними змінами.

3.7 Панель менеджера

Панель менеджера є важливим компонентом системи прокату автомобілів, де менеджер виступає в ролі центрального контролера замовлень (ДОДАТОК А – ManagerController.java). Ця панель дозволяє менеджеру ефективно керувати всіма аспектами замовлень, включаючи їх підтвердження, прийняття та відхилення.

Панель надає менеджеру повний огляд усіх нових замовлень, що надійшли в систему. Він може переглянути деталі кожної роботи, такі як дата, тип транспортного засобу та особливі вимоги клієнта, за наявності.

Управління замовленнями здійснюється через панель, де менеджер має можливість затвердити замовлення, якщо всі умови відповідають вимогам компанії. Прийняті замовлення надсилаються клієнту з підтвердженням. Якщо замовлення не відповідає певним критеріям або має конфлікти з іншими замовленнями, менеджер має право відхилити його і повідомити клієнту причину.

3.8 Панель користувача

Панель користувача є важливою складовою веб-додатку, де користувач має доступ до різноманітних функцій та інформації. Ця панель надає користувачам зручний інтерфейс, де вони можуть вибрати автомобіль для оренди, ознайомитися з головною сторінкою сайту, а також відвідати сторінку "Про нас" з короткою інформацією про компанію (ДОДАТОК А – HomeController.java, RentController.java).

Панель користувача також надає особистий кабінет, де зберігається вся інформація про користувача (ДОДАТОК А – ProfileController.java). Сюди входять персональні дані, контактна інформація, історія замовлень і статус замовлень.

В обліковому записі користувача також є можливість поповнити рахунок, що дозволяє користувачеві оплачувати оренду автомобіля зручним для нього способом. Користувач може переглядати свої замовлення, і якщо замовлення підтверджено, користувач має можливість повернути замовлення, якщо його плани змінилися або виникли непередбачувані обставини. Якщо замовлення відхилено і причина відхилення вказана в письмовій формі, користувач може ознайомитися з нею для подальшого вирішення проблем або внесення необхідних змін.

3.9 Алгоритм роботи

Адміністратор

Перед початком своєї роботи, адміністратор авторизується у свій обліковий запис. Після успішної авторизації, його перенаправляє на сторінку адміністратора. На даній сторінці доступні 3 списки: автомобілів, користувачів та менеджерів (рис. 3.3).

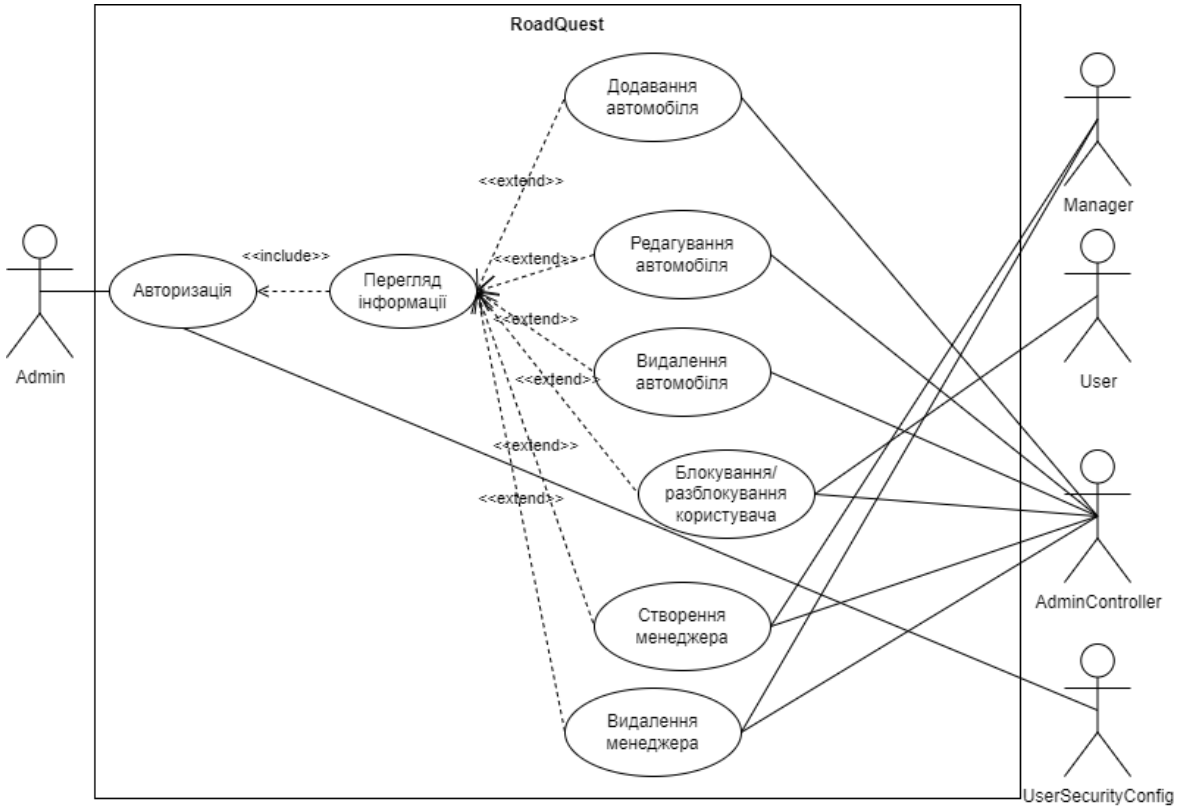

Рисунок 3.3 – Admin Use Case

В першому списку є доступ до додання нового автомобіля до списку, редагування та видалення наявного. При натисканні на кнопку «Add car», адміністратора перенесе на сторінку додання нового автомобіля з певними полями до заповнення. Як поля були заповнені і натиснута кнопка «Save», то адміністратора повертає на його сторінку з оновленим списком. Такий самий процес для редагування інформації по натиску кнопки «Update». Якщо адміністратор бажає видалити автомобіль, то натискає «Delete» навпроти потрібної, сторінка оновлюється і отримує новий список автомобілів.

В другому списку, списку користувачів, зберігається коротка інформація про всіх клієнтів. Якщо клієнт має активний статус, то відображається кнопка «Block» і при натисканні її, користувачу блокується доступ до сервісу. Інакше – кнопка «Unlock» і відновляє доступ до сервісу.

В третьому списку (списку менеджерів) відображається інформація про нинішніх менеджерів сервісу. При натисканні на кнопку «Add manager» відкривається сторінка додання менеджеру з певними полями. При натиску кнопки «Save», адміністратора повертає на свою сторінку з оновленим списком. Якщо натиснути кнопку «Delete» навпроти менеджера, видалить його із наявного списку і забороняє менеджеру дозвіл до системи.

Менеджер

Перед початком своєї роботи, менеджер авторизується у свій обліковий запис. Після успішної авторизації, його перенаправляє на свою сторінку. На даній сторінці доступний список замовлень (рис. 3.4).

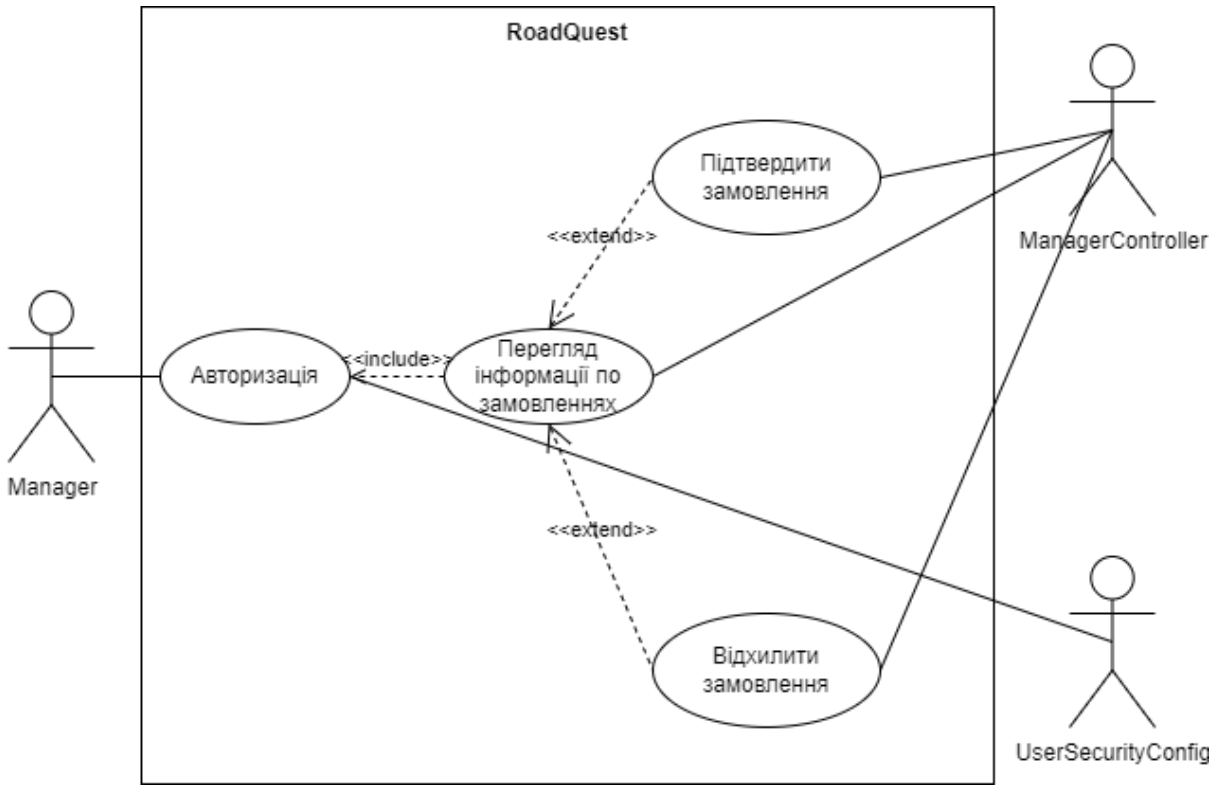

Рисунок 3.4 – Manager Use Case

- Якщо замовлення статус «New», менеджер приймає замовлення, або відхиляє його вписавши причину.
- Якщо замовлення статус «On hold», менеджер чекає на оплату замовлення від клієнту.
- Якщо замовлення статус «Paid», менеджер чекає на повернення автомобіля.
- Якщо замовлення статус «Completed», відображає що оренда закінчилась позитивно і значення дати коли було завершено замовлення заноситься до потрібної сутності бази даних, яка за допомоги тригеру видалить значення цього замовлення через 7 днів після його закриття.
- Якщо замовлення статус «Refunded», така сама логіка як і з попереднім пунктом. Проте відображає, що замовлення було відхилене менеджером.

Користувач

Як звичайний клієнт, користувач, до/після авторизації може переглядати сторінки «Home» та «About» (рис. 3.5). На першій сторінці знаходить загальна інформація про сайт та компанію з кнопкою «Choose a car», яка перенаправляє на сторінку замовлення (тільки авторизованих користувачів). Сторінка «About» дозволяє переглянути додаткову інформацію про компанію. Також доступні дві сторінки: авторизації та реєстрації, у випадку якщо немає облікового запису.

Після успішного входу до акаунту, у користувача з’являється змога орендувати автомобіль. По натиску кнопки на головній сторінці, або в шапці сайту, клієнт переходить на сторінку доступних автомобілів. На даній сторінці, присутня пагінація сторінок, для переходу між списками доступних машин. Окрім цього ліворуч списку знаходиться блок фільтрів. Серед цих критеріїв знаходяться типи автомобілів: Family, Soft, Off-road та Super. При обрання

критерію та натисканню кнопки «Find», сторінка оновлюється і повертає відфільтрований список автомобілів.

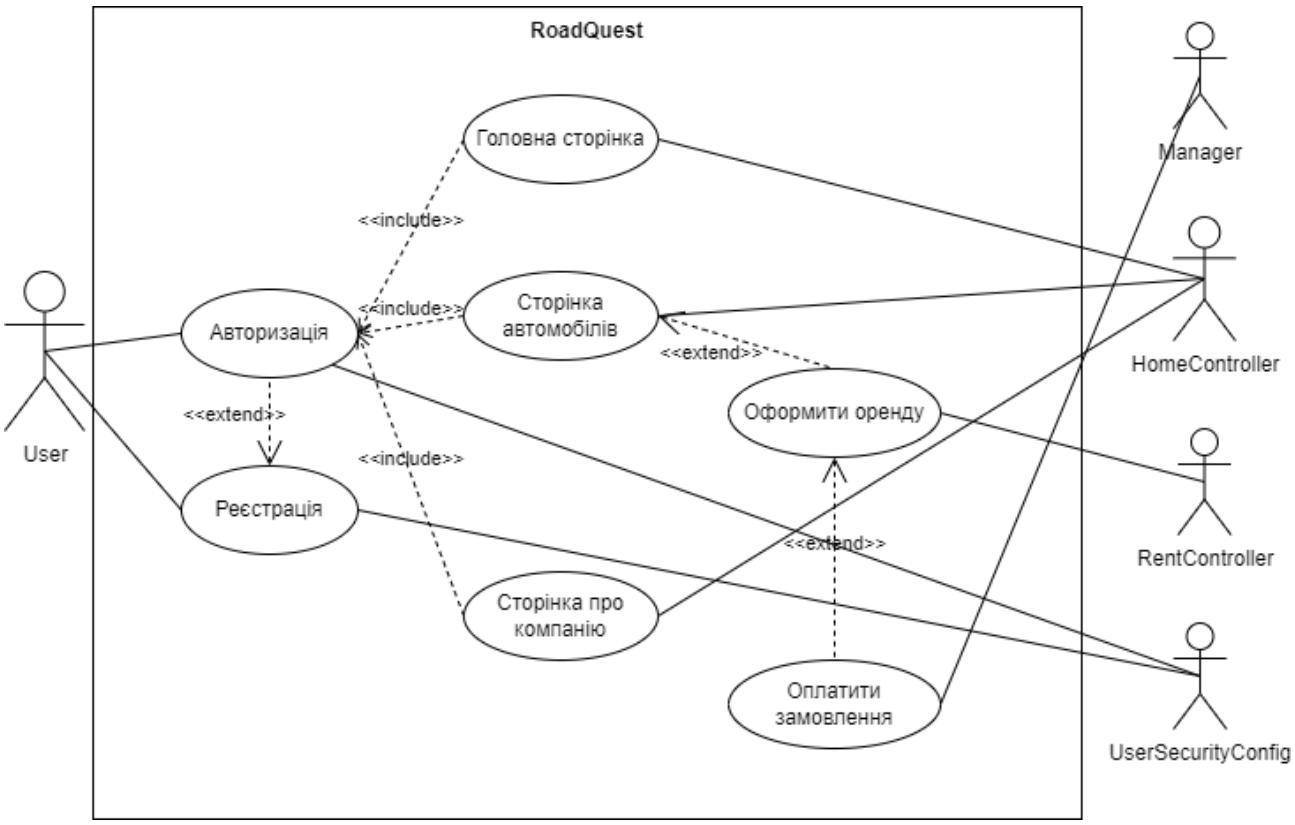

Рисунок 3.5 – User Use Case

Після замовлення автомобіля, заповнивши відповідні поля, користувача пересилає на профіль, де знаходиться його основа інформація і список актуальних замовлень. Одразу після оформлення оренди, замовлення має статус «New» і очікує підтвердження менеджера. Потім стає «On hold» і очікує оплати від клієнту. Натиснувши кнопку «Pay», замовлення оплачується і статус стає «Paid». Тоді з’являється кнопка «Return», натиснувши яку, можна повернути автомобіль і закрити оренду отримуючи статус «Completed». Якщо ж менеджер відхилив замовлення, статус стає «Refunded» і клієнт може переглянути повідомлення менеджера натиснувши кнопку «Show description».

Більш детальну інформацію про програмну реалізацію можна знайти на моєму GitHub репозиторії.

ВИСНОВКИ

В процесі роботи, було створено веб-додаток для автоматизація оренди автомобілів «RoadQuest».

У ході виконання кваліфікаційної роботи були виконані такі завдання:

- проведено аналіз сучасних джерел, присвячених розробці веб-додатків, з метою створення постановки задач для подальшого виконання;
- розглянуто наявні інструменти для розробки веб-додатків та обрано ті, що допоможуть досягти максимальної зручності та простоти використання, а також забезпечать відсутність недоліків у проектуванні власного додатку;
- були проаналізовані основні інструменти, які використовуються для створення веб-додатків;
- впроваджений якісний дизайн веб додатку;
- реалізовано збереження інформації про клієнтів та надані послуги;
- визначення бізнес-ідеї;
- захист додатку від втрати інформації;
- розмежування ролей доступу під час користування додатком;
- залучення бази даних до збереження даних в додатку.

Розроблений додаток здатний до розширення функціоналу. До такого функціоналу відноситься підключення платіжного шлюзу, через сторонні сервіси. Відстеження машини за допомоги GPS та повернення машини назад до компанії через після підтвердження від менеджера (у разі пошкодження автомобіля, користувач повинен відшкодувати збитки). Одним із варіантів покращення функціоналу є впровадження штучного інтелекту до пошуку потрібного автомобіля, створення мобільного додатку та чат боту в телеграм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is ICT (Information and Communications Technology)? [Electronic resource]. URL: https://www.techtarget.com/searchcio/definition/ICT-information-and-communications-technology-or-technologies (accessed: 13.05.2023).
2. Інформаційні й комунікаційні технології – UA5.org [Electronic resource]. URL: https://ua5.org/svit/281-nformacjjn-jj-komunkacjjn-tekhnolog.html (accessed: 13.05.2023).
3. Elveny M. et al. Web-based E-Commerce Products Grouping // J Phys Conf Ser. IOP Publishing Ltd, 2021. Vol. 1898, № 1.
4. Svobodová Z., Rajchlová J. Strategic behavior of e-commerce businesses in online industry of electronics from a customer perspective // Adm Sci. MDPI, 2020. Vol. 10, № 4.
5. The History and Evolution of eCommerce - Translate By Humans [Electronic resource]. URL: https://translatebyhumans.com/blog/history-and-evolution-of-ecommerce/ (accessed: 08.04.2023).
6. E-commerce evolution: Getting bigger and better every day [Electronic resource]. URL: https://www.the-future-of-commerce.com/2021/09/17/e-commerce-evolution/ (accessed: 08.04.2023).
7. Industry 4.0 [Electronic resource]. URL: https://www.it.ua/knowledge-base/technology-innovation/industry-4 (accessed: 18.03.2023).
8. What is the digital revolution? [Electronic resource]. URL: https://courses.minnalearn.com/en/courses/digital-revolution/the-digital-revolution/what-is-the-digital-revolution/ (accessed: 18.03.2023).
9. Rajamanickam M. et al. Fourth Industrial Revolution: Industry 4.0 // Integration of Mechanical and Manufacturing Engineering with IoT: a Digital Transformation. wiley, 2023. P. 41–84.
10. Ping L., Ying Z., Huang S. Role of information technology in the development of e-tourism marketing: A contextual suggestion // Econ Anal Policy. Elsevier BV, 2023.
11. Pathmanathan P.R. et al. The benefit and impact of E-commerce in tourism enterprises // 2021 2nd International Conference on Smart Computing and Electronic Enterprise: Ubiquitous, Adaptive, and Sustainable Computing Solutions for New Normal, ICSCEE 2021. Institute of Electrical and Electronics Engineers Inc., 2021. P. 193–198.
12. Le T.T.K. An Investigation into Customer Satisfaction of Customer Service Quality of an E-commerce Platform // ACM International Conference Proceeding Series. Association for Computing Machinery, 2022. P. 7–13.
13. Mai H.T. et al. Digital and Technological Solutions for Vietnam Tourism Services Development // Lecture Notes in Networks and Systems. Springer Science and Business Media Deutschland GmbH, 2023. Vol. 522. P. 515–522.

14. Karunakaran V., Sharma A. User Engagement Analysis of E-Commerce Websites from the Perspective of Eye Tracking. Institute of Electrical and Electronics Engineers (IEEE), 2023. P. 1–6.
15. Mitra D. et al. Importance of Coping with Cyber Security Challenges in E Commerce Business. Institute of Electrical and Electronics Engineers (IEEE), 2023. P. 1596–1601.
16. Malik S., Rana A., Bansal M. Improved Performance of Recommender System Based on Demographic Attributes // Lecture Notes on Data Engineering and Communications Technologies. Springer Science and Business Media Deutschland GmbH, 2021. Vol. 62. P. 83–92.
17. Arrioja-Castrejón E., López-Fernández A.M. Analysis of Medical Tourism and the Effect of Using Digital Tools to Profile Travelers in Mexico // Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. Springer Science and Business Media Deutschland GmbH, 2021. Vol. 393 LNICST. P. 143–161.
18. Dash P., Mishra J., Dara S. Sentiment Analysis on Social Network Data and Its Marketing Strategies: A Review // ECS Trans. The Electrochemical Society, 2022. Vol. 107, № 1. P. 7417–7425.
19. Vysotska V. Analytical Method for Social Network User Profile Textual Content Monitoring Based on the Key Performance Indicators of the Web Page and Posts Analysis // CEUR Workshop Proc. CEUR-WS, 2022. Vol. 3171. P. 1380–1402.
20. How to Build an Online Store in 2023: 9 Simple Steps [Electronic resource]. URL: https://www.websitebuilderexpert.com/building-online-stores/ (accessed: 08.04.2023).
21. How To Start an Online Store in 2023 (Step-by-Step Guide) [Electronic resource]. URL: https://www.shopify.com/blog/start-online-store#4 (accessed: 08.04.2023).
22. What is Web Application (Web Apps) and its Benefits [Electronic resource]. URL: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app (accessed: 18.03.2023).
23. Як перевірити швидкість завантаження сайту у новому PageSpeed Insights і зробити правильні висновки - Блог Arto Web Agency [Electronic resource]. URL: https://arto.agency/ua/check-website-speed-updated/ (accessed: 13.05.2023).
24. What Are The Best Web Development Tools in 2022? [Electronic resource]. URL: https://blog.hubspot.com/website/web-development-tools (accessed: 08.04.2023).
25. What is Scrum? | Scrum.org [Electronic resource]. URL: https://www.scrum.org/resources/what-scrum-module (accessed: 13.05.2023).
26. What is Agile? | Agile 101 | Agile Alliance [Electronic resource]. URL: https://www.agilealliance.org/agile101/ (accessed: 13.05.2023).

27. 10 popular database management systems (DBMS) [List] [Electronic resource]. URL: https://www.stackscale.com/blog/popular-database-management-systems/ (accessed: 08.04.2023).
28. MySQL :: MySQL Workbench [Electronic resource]. URL: https://www.mysql.com/products/workbench/ (accessed: 08.04.2023).
29. What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF Example [Electronic resource]. URL: https://www.guru99.com/database-normalization.html (accessed: 08.04.2023).
30. What's a design pattern? [Electronic resource]. URL: https://refactoring.guru/design-patterns/what-is-pattern (accessed: 08.04.2023).
31. What is API: Definition, Specifications, Types, Documentation | AltexSoft [Electronic resource]. URL: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/ (accessed: 08.04.2023).
32. IntelliJ IDEA – the Leading Java and Kotlin IDE [Electronic resource]. URL: https://www.jetbrains.com/idea/ (accessed: 20.05.2023).
33. What is an IDE? [Electronic resource]. URL: https://www.redhat.com/en/topics/middleware/what-is-ide (accessed: 08.04.2023).
34. WebStorm: The Smartest JavaScript IDE, by JetBrains [Electronic resource]. URL: https://www.jetbrains.com/webstorm/ (accessed: 20.05.2023).
35. Apache Tomcat® - Welcome! [Electronic resource]. URL: https://tomcat.apache.org/ (accessed: 08.04.2023).
36. Figma: the collaborative interface design tool. [Electronic resource]. URL: https://www.figma.com/ (accessed: 08.04.2023).
37. What is JDBC? - IBM Documentation [Electronic resource]. URL: https://www.ibm.com/docs/en/informix-servers/12.10?topic=started-what-is-jdbc (accessed: 08.04.2023).
38. Папіжук Д. О., Шовкопляс О. А. Веб-додаток для організації спільного використання автомобілів з метою автоматизації в туризмі. Суми, 2023. 98 р.
39. BlaBlaCar - пошук надійних попутників | BlaBlaCar [Electronic resource]. URL: https://www.blablacar.com.ua/ (accessed: 08.04.2023).

ДОДАТОК А
ЛІСТИНГ ПРОГРАМНОГО КОДУ

A1 UserSecurityConfig.java

```
@Configuration
@RequiredArgsConstructor
public class UserSecurityConfig {

    private static final String[] WHITE_LIST = {
            "/css/**",
            "/img/**",
            "/signup",
            "/",
            "/about",
            "index.html"
    };

    private static final String[] ADMIN_LIST = {
            "/admin",
            "/admin/addCar",
            "/admin/editCar",
            "/admin/saveCar",
            "/admin/deleteCar",
            "/admin/changeUser",
            "/admin/addManager",
            "/admin/saveManager",
            "/admin/deleteManager",
            "/admin/changePrice"
    };

    private static final String[] RESOURCE_LIST = {
            "/admin/**",
            "/manager/**",
            "/rent/**",
            "/profile/**"
    };

    private static final String[] MANAGER_LIST = {
            "/manager",
            "/manager/inspectOrder",
            "/manager/declineApplication",
            "/manager/acceptApplication"
    };
    private final UserDetailsServiceImpl service;

    @Bean
    public PasswordEncoder encoder() {
        return new BCryptPasswordEncoder();
    }
```

```
    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider daoAuthenticationProvider = new
DaoAuthenticationProvider();
        daoAuthenticationProvider.setPasswordEncoder(encoder());
        daoAuthenticationProvider.setUserDetailsService(service);
        return daoAuthenticationProvider;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http)
throws Exception {
        http.formLogin()
                .loginPage("/login")
                .successHandler(authenticationSuccessHandler())
                .failureForwardUrl("/login?error=true")
                .permitAll()
                .and()
                .logout(logout ->
logout.logoutSuccessUrl("/").permitAll());

        http.authorizeHttpRequests(configurer ->
                configurer

.requestMatchers(MANAGER_LIST).hasAuthority("MANAGER")

.requestMatchers(ADMIN_LIST).hasAuthority("ADMIN")
                        .requestMatchers(WHITE_LIST).permitAll()
                        .anyRequest().authenticated()
        );
        http.httpBasic();
        http.csrf().disable();
        return http.build();
    }

    @Bean
    public AuthenticationSuccessHandler
authenticationSuccessHandler() {
        return new RoleAuthenticationSuccessHandler();
    }

    @Bean
    public WebMvcConfigurer webMvcConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void
addResourceHandlers(ResourceHandlerRegistry registry) {
                registry
                        .addResourceHandler(RESOURCE_LIST)

.addResourceLocations("classpath:/static/");
```

```
            }
        };
    }
}
```

A2 LoginController.java

```
@Controller
public class LoginController {

    @GetMapping("/login")
    public String login() {
        return RQLiterals.LOGIN_PAGE.value();
    }

    @GetMapping("/logout")
    public String logout() {
        return RQLiterals.REDIRECT_LOGOUT_PAGE.value();
    }
}
```

A3 SignUpController.java

```
@Controller
@RequiredArgsConstructor
public class SignUpController {

    private final UserRQServiceImpl service;
    private final PasswordEncoder encoder;

    @GetMapping("/signup")
    public String createNewUser(UserModel userModel, Model model)
{
        model.addAttribute("userModel", userModel);
        return RQLiterals.SIGNUP_PAGE.value();
    }

    @PostMapping("/signup")
    public String createNewUser(@Valid
@ModelAttribute("userModel") UserModel userModel,
                                BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return RQLiterals.SIGNUP_PAGE.value();
        }
        service.save(
                User.builder()
                        .login(userModel.getLogin())

.password(encoder.encode(userModel.getPassword()))
                        .email(userModel.getEmail())
                        .firstName(userModel.getFirstName())
                        .lastName(userModel.getLastName())
                        .phoneNumber(userModel.getPhoneNumber())
                        .role(new Role(3, "BASIC"))
```

```
                        .status(true)
                        .build()
        );
        return RQLiterals.REDIRECT_SIGNUP_PAGE.value();
    }

}
```

A4 AdminController.java

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/admin")
public class AdminController {

    private static final Role BASIC = new Role(3, "BASIC");
    private static final Role MANAGER = new Role(2, "MANAGER");
    private final CarRQServiceImpl carService;
    private final CarTypeRQServiceImpl carTypeService;
    private final UserRQServiceImpl userService;
    private final PasswordEncoder encoder;

    @GetMapping
    public String init(Model model) {
        model.addAttribute("cars", carService.findAll())
                .addAttribute("users",
userService.findByRole(BASIC))
                .addAttribute("managers",
userService.findByRole(MANAGER))
                .addAttribute("types", carTypeService.findAll());
        return RQLiterals.ADMIN_PAGE.value();
    }

    @GetMapping("/addCar")
    public String addCar(Model model) {
        Car newCar = new Car();

        model.addAttribute("car", newCar);
        model.addAttribute("types", carTypeService.findAll());
        return RQLiterals.CAR_PAGE.value();
    }

    @GetMapping("/editCar")
    public String editCar(@RequestParam("carId") int id, Model
model) {
        model.addAttribute("car", carService.findById(id));
        model.addAttribute("types", carTypeService.findAll());
        return RQLiterals.CAR_PAGE.value();
    }

    @PostMapping("/saveCar")
    public String saveCar(@Valid @ModelAttribute("car") CarModel
```

```
carModel,
                            BindingResult bindingResult, Model
model) {
        if (bindingResult.hasErrors()) {
            model.addAttribute("types", carTypeService.findAll());
            return RQLiterals.CAR_PAGE.value();
        }
        carService.save(
                Car.builder()
                        .id(carModel.getId())
                        .name(carModel.getName())
                        .carType(carModel.getCarType())
                        .status(true)
                        .build()
        );

        return RQLiterals.REDIRECT_ADMIN.value();
    }

    @GetMapping("/deleteCar")
    public String deleteCar(@RequestParam("carId") int id) {
        carService.deleteById(id);

        return RQLiterals.REDIRECT_ADMIN.value();
    }

    @GetMapping("/changeUser")
    public String changeUserStatus(@RequestParam("userId") int id)
{
        User userById = userService.findById(id);
        userById.setStatus(!userById.isStatus());

        userService.save(userById);
        return RQLiterals.REDIRECT_ADMIN.value();
    }

    @GetMapping("/addManager")
    public String addManager(Model model) {
        User manager = new User();

        model.addAttribute("manager", manager);
        return RQLiterals.CREATE_MANAGER_PAGE.value();
    }

    @PostMapping("/saveManager")
    public String saveManager(@Valid @ModelAttribute("manager")
UserModel managerModel,
                              BindingResult bindingResult, Model
model) {
        if (bindingResult.hasErrors()) {
            return RQLiterals.CREATE_MANAGER_PAGE.value();
```

```
        }
        userService.save(
                User.builder()
                        .login(managerModel.getLogin())

.password(encoder.encode(managerModel.getPassword()))
                        .email(managerModel.getEmail())
                        .firstName(managerModel.getFirstName())
                        .lastName(managerModel.getLastName())

.phoneNumber(managerModel.getPhoneNumber())
                        .role(MANAGER)
                        .status(true)
                        .build()
        );

        return RQLiterals.REDIRECT_ADMIN.value();
    }

    @GetMapping("/deleteManager")
    public String deleteManager(@RequestParam("managerId") int id)
{
        userService.deleteById(id);

        return RQLiterals.REDIRECT_ADMIN.value();
    }

    @GetMapping("/changePrice")
    public String changeCarTypePrice(@RequestParam("typeId") int
id, @RequestParam("price") int price) {
        CarType carType = carTypeService.findById(id);
        carTypeService.save(CarType.builder()
                .id(carType.getId())
                .name(carType.getName())
                .price(price).build());
        return RQLiterals.REDIRECT_ADMIN.value();
    }
}
```

A5 ManagerController.java

```
@Controller
@RequestMapping("/manager")
@RequiredArgsConstructor
public class ManagerController {

    private static final RentStatus ON_HOLD = new RentStatus(2,
"On hold");
    private static final RentStatus COMPLETED = new RentStatus(4,
"Completed");
    private static final RentStatus REFUNDED = new RentStatus(5,
"Refunded");
```

```
    private final ApplicationRQServiceImpl service;
    private final CarRQServiceImpl carService;

    @GetMapping
    public String init(Model model) {
        model.addAttribute("applications", service.findAll());
        return RQLiterals.MANAGER_PAGE.value();
    }

    @GetMapping("/inspectOrder")
    public String inspectOrder(@RequestParam("applicationId") int
id, Model model) {
        Application application = service.findById(id);
        model.addAttribute("order", application);
        System.out.println(application.getStartDate());
        System.out.println(application.getEndDate());
        return RQLiterals.INSPECT_ORDER.value();
    }

    @PostMapping("/declineApplication")
    public String
declineApplication(@ModelAttribute("application") ApplicationModel
applicationModel) {
        int id = applicationModel.getId();
        Application application = service.findById(id);
        Car car = application.getCar();


application.setDescription(applicationModel.getDescription());
        application.setRentStatus(REFUNDED);
        car.setStatus(true);

        carService.save(car);
        service.save(application);
        return RQLiterals.REDIRECT_MANAGER.value();
    }

    @GetMapping("/acceptApplication")
    public String acceptApplication(@RequestParam("applicationId")
int id) {
        Application application = service.findById(id);
        application.setRentStatus(ON_HOLD);
        service.save(application);

        return RQLiterals.REDIRECT_MANAGER.value();
    }
}
```

A6 ProfileController.java

```
@Controller
@RequestMapping("/profile")
```

```
@RequiredArgsConstructor
public class ProfileController {

    private static final RentStatus PAID = new RentStatus(3,
"Paid");

    private static final RentStatus COMPLETED = new RentStatus(4,
"Completed");
    private final UserRQServiceImpl userService;
    private final ApplicationRQService applicationService;
    private final CarRQServiceImpl carService;

    @PostMapping("/cashIn")
    public String cashIn(@RequestParam("userId") int id,
@RequestParam("cash") int cash) {
        User byId = userService.findById(id);
        byId.setBill(byId.getBill() + cash);
        userService.save(byId);

        return RQLiterals.REDIRECT_USER.value();
    }

    @GetMapping("/payApplication")
    public String payApplication(@RequestParam("orderId") int id)
{
        Application application = applicationService.findById(id);
        User customer = application.getCustomer();
        customer.setBill(customer.getBill() -
application.getPrice());
        application.setRentStatus(PAID);

        userService.save(customer);
        applicationService.save(application);

        return RQLiterals.REDIRECT_USER.value();
    }

    @GetMapping("/returnCar")
    public String returnCar(@RequestParam("orderId") int id) {
        Application application = applicationService.findById(id);
        Car car = application.getCar();

        application.setRentStatus(COMPLETED);
        car.setStatus(true);

        applicationService.save(application);
        carService.save(car);

        return RQLiterals.REDIRECT_USER.value();
    }
}
```

A7 HomeController.java

```
@Controller
@RequiredArgsConstructor
public class HomeController {

    private final UserRQServiceImpl userService;
    private final CarRQService carService;
    private final ApplicationRQService applicationService;
    private final CarTypeRQService carTypeService;
    private List<CarType> selectedFilters;
    private String filteredUrl = "";

    @GetMapping("/profile")
    public String profile(@AuthenticationPrincipal UserDetails
userDetails,
                          Model model, Authentication auth) {
        Collection<? extends GrantedAuthority> authorities =
auth.getAuthorities();
        for (final GrantedAuthority grantedAuthority :
authorities) {
            String authorityName =
grantedAuthority.getAuthority();
            if (authorityName.equals("ADMIN")) {
                return RQLiterals.REDIRECT_ADMIN.value();
            } else if (authorityName.equals("MANAGER")) {
                return RQLiterals.REDIRECT_MANAGER.value();
            }
        }

        initUserCabinet(userDetails, model);
        return RQLiterals.USER_PROFILE.value();
    }

    @GetMapping("/rent")
    public String showRentPage(@RequestParam(required = false)
                               List<CarType> filters,
                               @RequestParam(required = false,
defaultValue = "0")
                               int page,
                               @RequestParam(required = false,
defaultValue = "10")
                               int size,
                               Model model, HttpServletRequest
request) {

        Pageable pageable = PageRequest.of(page, size);
        Page<Car> cars;
        Page<Car> pages;
        List<CarType> carTypes;
```

```
        if (filters != null) {
            if
(request.getQueryString().matches("(?:filters=\\d+&?)+")) {
                setFilteredUrl(request.getQueryString() + "&");
            }
            setSelectedFilters(filters);
            cars = carService.findAllByCarTypeIn(filters,
pageable);
            pages = carService.findAllByCarTypeIn(filters,
pageable);
        } else {
            setFilteredUrl("");
            setSelectedFilters(new ArrayList<>());
            cars = carService.findAllActiveCars(pageable);
            pages = carService.findAllActiveCars(pageable);
        }

        model.addAttribute("actualCars", cars);
        model.addAttribute("pages", pages);
        model.addAttribute("filters", carTypeService.findAll());
        model.addAttribute("selectedFilters",
getSelectedFilters());
        model.addAttribute("filteredUrl", getFilteredUrl());
        return RQLiterals.RENT_PAGE.value();
    }

    @GetMapping("/about")
    public String showAboutPage(Model model) {
        return RQLiterals.ABOUT_PAGE.value();
    }

    private void initUserCabinet(UserDetails userDetails, Model
model) {
        String login = userDetails.getUsername();
        User user = userService.findByLogin(login);
        model.addAttribute("currentUser", user);
        model.addAttribute("orders",
applicationService.findByCustomer(user));
    }

    public List<CarType> getSelectedFilters() {
        return selectedFilters;
    }

    public void setSelectedFilters(List<CarType> selectedFilters)
{
        this.selectedFilters = selectedFilters;
    }

    public String getFilteredUrl() {
        return filteredUrl;
```

```
    }

    public void setFilteredUrl(String filteredUrl) {
        this.filteredUrl = filteredUrl;
    }
}
```

A8 RentController.java

```
@Controller
@RequestMapping("/rent")
@RequiredArgsConstructor
public class RentController {

    private static final RentStatus NEW = new RentStatus(1,
"New");
    private final UserRQServiceImpl userService;
    private final CarRQServiceImpl carService;
    private final ApplicationRQService applicationService;

    @GetMapping("/fillApplication")
    private String fillApplication(@RequestParam("carId") int id,
                                   @AuthenticationPrincipal
UserDetails userDetails,
                                   Model model, Authentication
auth) {

        String login = userDetails.getUsername();
        model.addAttribute("chosenCar", carService.findById(id))
                .addAttribute("user",
userService.findByLogin(login))
                .addAttribute("application", new Application());
        return RQLiterals.FILL_APPLICATION_PAGE.value();
    }

    @PostMapping("/saveApplication")
    private String saveApplication(@Valid
@ModelAttribute("application")
                                   ApplicationModel
applicationModel,
                                   BindingResult bindingResult,
                                   @RequestParam(value =
"passport", required = false) String passport) {
        if (bindingResult.hasErrors()) {
            return RQLiterals.FILL_APPLICATION_PAGE.value();
        }

        if (passport != null) {
            User user =
userService.findById(applicationModel.getCustomer().getId());
            user.setPassport(Integer.valueOf(passport));
            userService.save(user);
```

```
        }

        Car car = applicationModel.getCar();
        car.setStatus(false);

        applicationService.save(
                Application.builder()
                        .id(applicationModel.getId())
                        .customer(applicationModel.getCustomer())
                        .car(car)

.startDate(applicationModel.getStartDate())
                        .endDate(applicationModel.getEndDate())
                        .rentStatus(NEW)

.price(applicationModel.getCar().getCarType().getPrice()).build()
        );
        carService.save(car);

        return RQLiterals.REDIRECT_USER.value();
    }
}
```

ДОДАТОК Б
ЛІСТИНГ БІБЛІОТЕК

```
<dependencies>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-data-
jpa</artifactId>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-
thymeleaf</artifactId>
          </dependency>
          <dependency>
               <groupId>org.thymeleaf.extras</groupId>
               <artifactId>thymeleaf-extras-
springsecurity6</artifactId>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-
security</artifactId>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-
validation</artifactId>
          </dependency>
          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-web</artifactId>
          </dependency>

          <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-devtools</artifactId>
               <scope>runtime</scope>
               <optional>true</optional>
          </dependency>
          <dependency>
               <groupId>com.mysql</groupId>
               <artifactId>mysql-connector-j</artifactId>
               <scope>runtime</scope>
          </dependency>
          <dependency>
               <groupId>org.projectlombok</groupId>
               <artifactId>lombok</artifactId>
               <optional>true</optional>
          </dependency>
          <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```