МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Вебзастосунок продажу автомобілів

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Олексій ІВАНОВ**

«__» _____ 2026 р.

Керівник роботи
канд. техн. наук,
доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 2026 р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Іванов Олексій

1. Тема кваліфікаційної роботи вебзастосунок продажу автомобілів затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2025 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Очікуваним результатом є створення вебзастосунку для забезпечення багатокористувацького доступу до товарів та послуг з продажу авто.
4. Перелік питань, що підлягають розробці:

~ провести аналіз предметної області та існуючих аналогів вебзастосунків продажу авто;

~ проаналізувати програмні та інструментальні засоби для розробки інтернет-магазину;

˜ розробити логіку бази даних;

˜ розробити логіку вебзастосунку;

– виконати програмну реалізацію та тестування інтернет-магазину.

5. Перелік графічних матеріалів:

– презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «27» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок продажу автомобілів

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на створення вебзастосунку продажу автомобілів	26.12.2025	29.12.2025	Виконано
2.	Огляд літератури за обраною темою	05.01.2026	19.01.2026	Виконано
3.	Складання календарного плану КБР	20.01.2026	21.01.2026	Виконано
4.	Аналіз аналогічних вебзастосунків	22.01.2026	28.01.2026	Виконано
5.	Проєктування схеми бази даних та вибір стеку технологій	29.01.2026	16.02.2026	Виконано
6.	Моделювання та конструювання вебзастосунку, розробка бази даних	17.02.2026	17.03.2026	Виконано
7.	Кодування функціоналу, перевірка роботи вебзастосунку, тестування безпеки	18.03.2026	17.04.2026	Виконано
8.	Отримання відгуку керівника КБР	20.04.2026	22.04.2026	Виконано
9.	Оформлення КБР та презентації	23.04.2026	15.05.2026	Виконано
10.	Попередній захист	27.05.2026	27.05.2026	Виконано
11.	Завершення оформлення КБР та презентації	28.05.2026	29.05.2026	Виконано
12.	Рецензування	06.06.2026	06.06.2026	Виконано
13.	Захист кваліфікаційної роботи	22.06.2026	22.06.2026	Виконано

Здобувач _____ **Олексій ІВАНОВ**

«__» _____ 2026 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок продажу автомобілів

Здобувач 409 гр.: Іванов Олексій

Керівник: канд. техн. наук, доцент Давиденко Євген

Актуальність теми даної роботи обумовлена розвитком сучасних технологій та змінами у виборі користувачів у сфері комерції. В наш час цифрових технологій багато сфер життєдіяльності людини перейшло в інтернет-простір через зручність його використання. Він дозволяє користувачам економити власний час та надає зручний і швидкий інструмент для пошуку та придбання товарів.

Вебзастосунки дають змогу масштабувати бізнес, оптимізувати різні процеси, і надають споживачу зручність та можливість отримання багатьох послуг онлайн з будь-якої точки світу.

Робота присвячена розробці вебзастосунку інтернет-магазину автомобілів, який надає користувачам зручний та ефективний інструмент для пошуку та придбання автомобілів онлайн.

Метою кваліфікаційної роботи є розробка вебзастосунку для забезпечення багатокористувацького доступу до товарів та послуг з продажу авто.

Об’єктом роботи процес автоматизації продажу авто.

Предметом роботи є програмні засоби автоматизації продажу авто.

Кваліфікаційна роботи бакалавра включає вступ, чотири розділи, висновки та перелік джерел посилання.

У вступі розкривається актуальність обраної теми, описується мета, надається короткий огляд поставленої задачі, та надається опис об’єкту, предмету та методів дослідження.

У першому розділі досліджується предметна область, проводиться аналіз існуючих аналогів та розглядаються особливості їх роботи. В результаті цього аналізу сформовано завдання для виконання кваліфікаційної бакалаврської роботи.

У другому розділі розглядаються та аналізуються існуючі технології, підходи та інструменти для проєктування інтернет-магазину автомобілів. Також

формується специфікація вимог до ПЗ, яка представляє собою повний опис поведінки програмного забезпечення, що розробляється.

У третьому розділі розглядається стек технологій для проєктування інтернет-магазину автомобілів. Описується конструювання програмного забезпечення та технології розробки. Моделюється структура бази даних. Проєктується користувацький інтерфейс. Також в даному розділі представлено низку UML-діаграм, що описують роботу ПЗ.

У четвертому розділі розписується програмна реалізація, де описується клієнтська частина.

Кваліфікаційна робота викладена на 89 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 25 найменувань та 2 додатків. Праця містить 13 таблиць та 37 рисунків.

Ключові слова: вебзастосунок, інтернет-магазин автомобілів, клієнт, Angular, SQL Server.

ABSTRACT

of the Bachelor's Thesis

Car sales web application

Student of group 409: Ivanov Oleksii

Supervisor: candidate of technical Sciences, associate professor Davydenko Yevhen

The relevance of the topic of this work is due to the development of modern technologies and changes in the choice of users in the field of commerce. In our time of digital technologies, many areas of human life have moved into the Internet space due to the convenience of its use. It allows users to save their own time and provides a convenient and fast tool for searching and purchasing goods.

Web applications allow to scale business, optimize various processes, and provide the consumer with convenience and the ability to receive many services online from anywhere in the world.

The work is dedicated to the development of a web application for an online car store, which provides users with a convenient and effective tool for searching and purchasing cars online.

The purpose of the qualification work is to develop a web application to provide multi-user access to car sales products and services.

The object of the work is the process of automating car sales.

The subject of the work is software tools for automating car sales.

A bachelor's degree thesis includes an introduction, four sections, conclusions, and a list of references.

The introduction reveals the relevance of the chosen topic, describes the goal, provides a brief overview of the task, and provides a description of the object, subject and research methods.

The first section explores the subject area, analyzes existing analogues and considers the features of their work. As a result of this analysis, tasks for performing a qualifying bachelor's thesis are formed.

The second section considers and analyzes existing technologies, approaches and tools for designing an online car store. A software requirements specification is also formed, which is a complete description of the behavior of the software being developed.

The third section considers the technology stack for designing an online car store. The software design and development technologies are described. The database structure is modeled. The user interface is designed. This section also presents a number of UML diagrams describing the operation of the software.

The fourth section describes the software implementation, which describes the client part.

The qualification work is presented on 89 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 25 titles and 2 appendices. The work contains 13 tables and 37 illustrations.

Keywords: web application, online car store, client, Angular, SQL Server.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	Програмне зебезпечення
ПК	Персональний комп’ютер
СКБД	Система керування базами даних
ТОВ	Товариство з обмеженою відповідальністю

API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
DOM	Document Object Model
ER	Entity-Relationship
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDEF0	Integration Definition for Function Modeling
JS	JavaScript
JSON	JavaScript Object Notation
REST	Representational State Transfer
SPA	Single Page Application
SQL	Structured Query Language
SSL	Secure Sockets Layer
TS	TypeScript
T-SQL	Transact Structured Query Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
UTF	Unicode Transformation Format

ВСТУП

Актуальність обраної теми: у сучасному світі автомобіль доволі актуальна річ. У світі налічується близько 1,3 мільярда машин, і з кожним роком ця кількість збільшується. Очікується, що за поточний рік глобальні продажі нових автомобілів складуть 91,8 мільйона одиниць, що показує, що попит на легкові автомобілі, у 2026 році, залишиться на колишньому рівні. У свою чергу світове виробництво автомобілів збільшиться до 92,6 млн одиниць, що становить збільшення на 0,4% у річному обчисленні. Це показує, що авторинок нових авто зростає доволі непоганими темпами [1].

У наш час цифрових технологій багато сфер життєдіяльності людини перейшло в інтернет-простір і бізнес з продажу авто не є виключенням. Це дає змогу масштабувати бізнес, оптимізувати різні процеси, а споживач отримує економію свого часу, зручність та можливість отримання багатьох послуг онлайн з будь-якої точки світу. Тому розробка та вдосконалення інтернет-платформ є актуальним завданням для бізнесу в наші часи.

Практичне значення: такий застосунок створить зручну та ефективну платформу для ознайомлення з асортиментом авто, що пропонується компанією. Надання повноцінної та стислої інформації про товари допоможе покупцям зробити обдуманий вибір.

Мета роботи: розробка вебзастосунку для забезпечення багатокористувацького доступу до товарів та послуг з продажу авто.

Для досягнення мети необхідно вирішити наступні **завдання:**

– проаналізувати сферу продажів авто;

– проаналізувати програмні та інструментальні засоби для розробки інтернет-магазину;

– розробити логіку бази даних;

– розробити логіку вебзастосунку;

– виконати програмну реалізацію та тестування інтернет-магазину.

Об’єкт роботи: процес автоматизації продажу авто.

Предмет роботи: програмні засоби автоматизації продажу авто.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Аналіз предметної області та визначення об’єкта і предмета дослідження

Вебзастосунки для продажу автомобілів сьогодні відіграють важливу роль у сфері електронної комерції. Це пов’язано з тим, що все більше процесів купівлі та продажу транспортних засобів переходить в онлайн-середовище. Розвиток інформаційних технологій та доступність Інтернету зробили такі сервіси зручними для продавців і покупців.

Автомобільний ринок постійно змінюється, з’являються нові моделі, а попит зберігається як на нові, так і на вживані автомобілі. У цих умовах вебзастосунки дозволяють швидко знаходити необхідні пропозиції та спрощують їх порівняння і отримання детальної інформації.

Важливою особливістю сучасних вебзастосунків є їх орієнтація на користувача та забезпечення високого рівня інтерактивності. Це досягається за рахунок використання адаптивного дизайну, швидкої обробки запитів та інтеграції з зовнішніми сервісами [2]. Також, сучасні системи продажу автомобілів повинні враховувати такі фактори:

- безпека обробки персональних даних користувачів;
- надійність зберігання інформації;
- масштабованість системи;
- підтримка великої кількості одночасних користувачів.

Особливу роль відіграє якість користувацького інтерфейсу, тому що саме він забезпечує зручність взаємодії із системою. Інтуїтивно зрозумілий інтерфейс дозволяє зменшити час пошуку необхідної інформації та підвищити ефективність роботи. Користувачі онлайн-платформ продажу авто очікують швидкого доступу до інформації про товар, включаючи технічні характеристики, ціну, стан та історію обслуговування. Також великою перевагою є додаткові можливості, такі як отримання додаткових послуг, кредитування, страхування чи запис на тест-драйв. Таким чином, розробка вебзастосунку продажу автомобілів є актуальною задачею,

що потребує врахування сучасних підходів до побудови інформаційних систем та забезпечення високого рівня функціональності і зручності використання.

Саме тому об’єктом дослідження є процес організації онлайн-продажу автомобілів у вебсередовищі, а предметом дослідження є методи та програмні засоби розробки вебзастосунку, зокрема: пошук і фільтрація автомобілів, управління оголошеннями, взаємодія користувачів, обробка замовлень і платежів, а також інтеграція додаткових сервісів.

1.2 Аналіз сучасного стану програмних рішень

На сучасному ринку існує велика кількість вебзастосунків для продажу автомобілів, які можна поділити на дилерські вебзастосунки від офіційних виробників автомобілів та портали оголошень де можна розміщувати оголошення про автомобілі, що були у використанні.

Для аналізу використовувались такі критерії:

– функціональність;
– архітектура;
– переваги та недоліки.

Nissan (табл. 1.1, рис. 1.1) – японський автомобільний бренд, що пропонує широкий модельний ряд легкових авто та кросоверів [3]. Дилерські центри Nissan забезпечують продаж нових автомобілів, сервісне обслуговування та гарантійну підтримку, орієнтуючись на якість, інновації та комфорт для клієнтів.

Таблиця 1.1 – Офіційний дилер NISSAN

Назва	КИЙ АВТО – офіційний дилер Nissan.
Розробник	Кий Авто (офіційний дилер Nissan / ТОВ «Ніссан Мотор Україна»).
Архітектура	Вебзастосунок (клієнт-сервер).
Мова реалізації	HTML/CSS/JavaScript.

Кінець таблиці 1.1

Перелік функцій, характеристик	1) каталог нових моделей та детальні сторінки моделей (характеристики, ціни); 2) перелік автомобілів в наявності та прайс-листи; 3) запис на тест-драйв та контактна інформація; 4) фінансові пропозиції: кредити, лізинг, пропозиції Nissan Financial Services; 5) сервіс: запис на сервіс, оригінальні запчастини, гарантія, калькулятор технічного обслуговування; 6) trade-in та корпоративні пропозиції.
Переваги	1) офіційний дилер – довірчий статус, актуальні офіційні ціни; 2) інтеграція з сервісами виробника (конфігуратор, фінсервіси); 3) повний набір послуг (продаж, фінансування, сервіс).
Недоліки	1) орієнтований на нові авто від конкретного бренду; 2) недоступний для приватних оголошень та вторинного ринку.

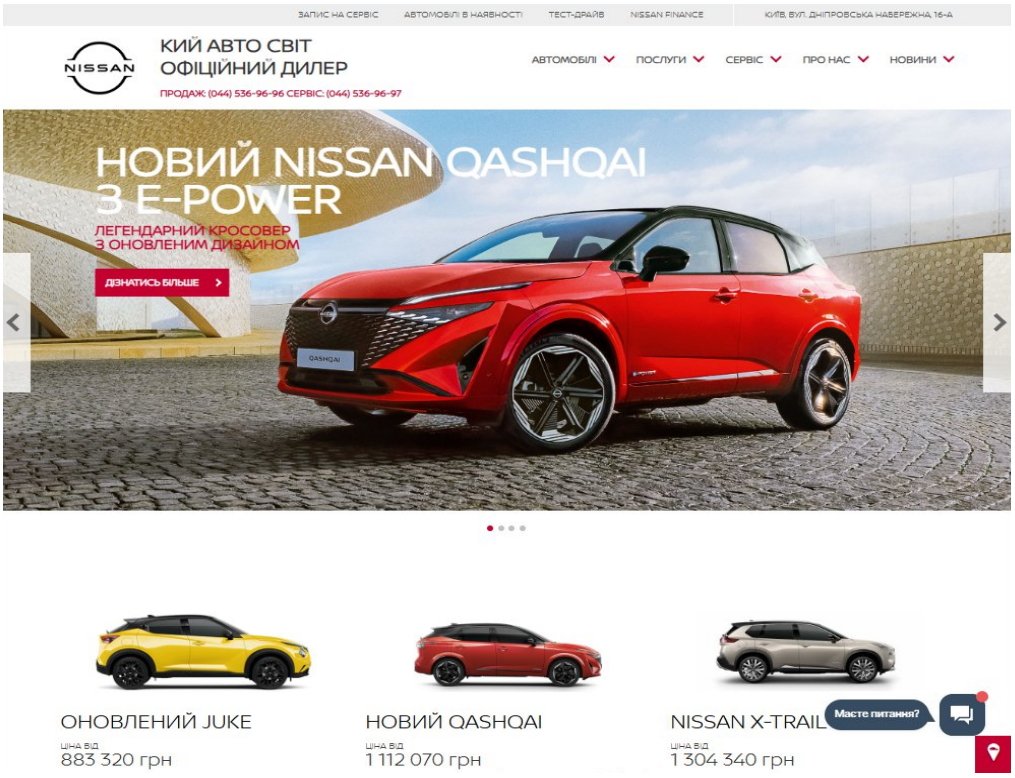

Рисунок 1.1 – Інтерфейс вебзастосунку офіційного дилера NISSAN

Саме так представлено вигляд вебзастосунку офіційного дилера NISSAN.

Renault (табл. 1.2, рис. 1.2) – французький автовиробник, відомий доступними та економічними автомобілями [4]. Офіційні дилери Renault пропонують продаж авто, фінансові програми, сервіс та технічну підтримку, роблячи акцент на практичності та сучасних технологіях.

Таблиця 1.2 – Офіційний дилер RENAULT

Назва	КИЙ АВТО – офіційний дилер Renault.
Розробник	Кий Авто / офіційний дилер Renault в Україні (моделі Renault).
Архітектура	Вебзастосунок (клієнт-сервер).
Мова реалізації	HTML/CSS/JavaScript.
Перелік функцій, характеристик	1) каталог моделей (легкові, кросовери, комерційні); 2) онлайн склад (автомобілі в наявності); 3) запит на придбання, тест-драйв, контактні форми; 4) фінансові інструменти: кредит, лізинг, страхування (Mobilize Financial Services); 5) сервіси: запис на сервіс, запчастини, гарантійні умови; 6) акції та новини дилера.
Переваги	1) повна підтримка бренду (офіційна інформація та актуальні пропозиції); 2) інструменти для фінансування і корпоративних клієнтів.
Недоліки	1) орієнтований на нові авто від конкретного бренду; 2) недоступний для приватних оголошень та вторинного ринку.

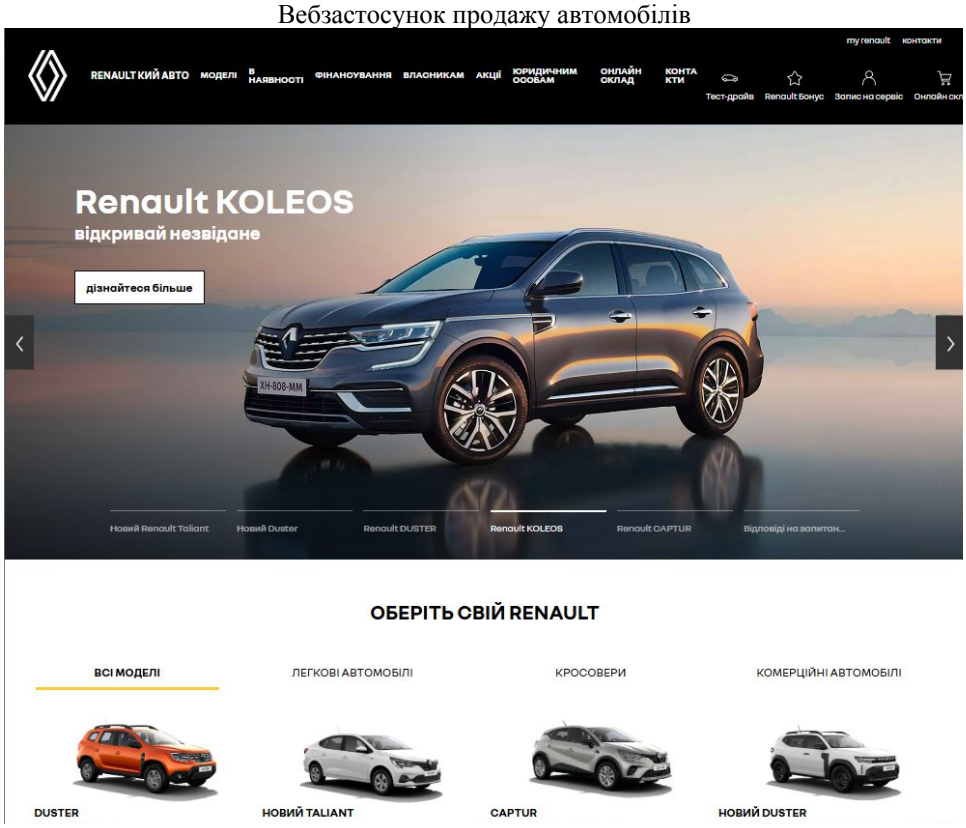

Рисунок 1.2 – Інтерфейс вебзастосунку офіційного дилера RENAULT

Таким чином представлено вигляд аналогічного вебзастосунку офіційного дилера RENAULT.

Toyota (табл. 1.3, рис. 1.3) – японський автомобільний бренд, відомий своєю надійністю та економічністю [5]. Офіційні дилери Toyota пропонують широкий вибір автомобілів, включаючи гібридні моделі, а також забезпечують сервісне обслуговування, гарантійну підтримку та фінансові рішення для клієнтів.

Таблиця 1.3 – Офіційний дилер TOYOTA

Назва	Тойота Центр – офіційний дилер Toyota.
Розробник	Офіційний дилер Toyota / Toyota Motor Corporation.
Архітектура	Вебзастосунок (клієнт-сервер).
Мова реалізації	HTML/CSS/JavaScript.

Кінець таблиці 1.3

Перелік функцій, характеристик	1) каталог нових автомобілів Toyota з детальними характеристиками та комплектаціями; 2) інформація про автомобілі в наявності та ціни; 3) онлайн-конфігуратор авто; 4) запис на тест-драйв та форма зворотного зв’язку; 5) фінансові послуги: кредитування, лізинг, страхування; 6) сервісне обслуговування: запис на сервіс, гарантія, оригінальні запчастини; 7) програми Trade-in та спеціальні пропозиції для бізнесу.
Переваги	1) офіційний статус – гарантія якості та достовірності інформації; 2) доступ до актуальних моделей і технологій (у тому числі гібридів); 3) повний цикл обслуговування: від купівлі до сервісу.
Недоліки	1) орієнтація лише на бренд Toyota; 2) відсутність вторинного ринку та приватних оголошень.

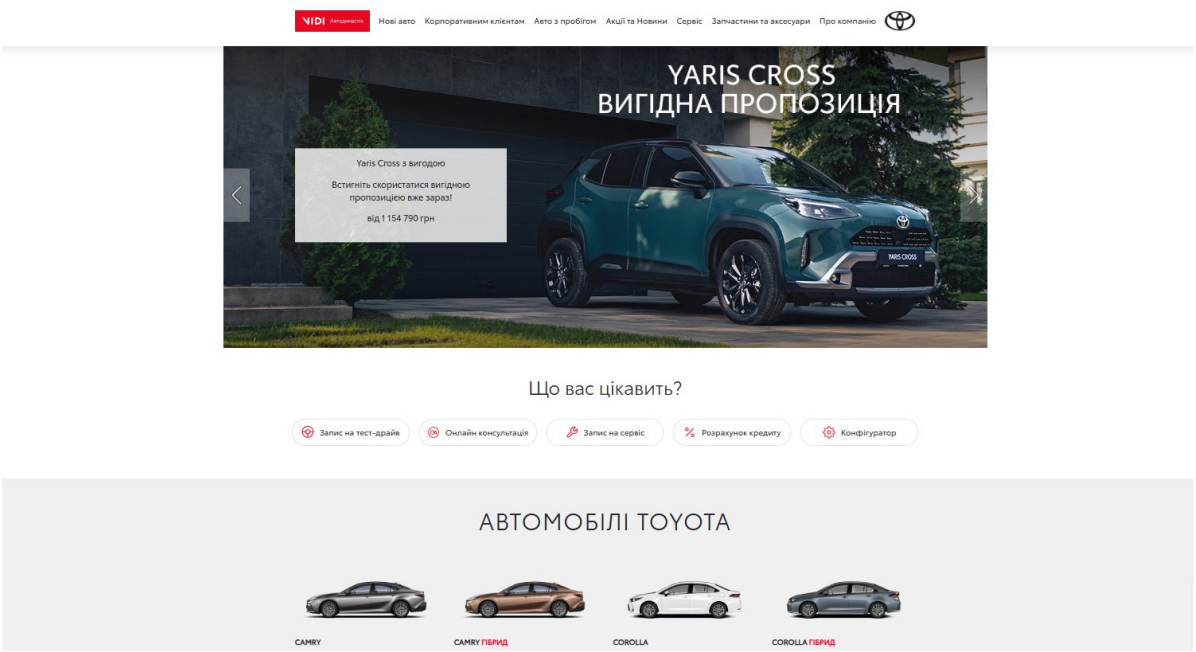

Рисунок 1.3 – Інтерфейс вебзастосунку офіційного дилера TOYOTA

Такий вигляд у аналогічного вебзастосунку офіційного дилера TOYOTA.

RST.ua (табл. 1.4, рис. 1.4) – український онлайн-портал оголошень, спеціалізований на купівлі та продажу автомобілів [6]. Платформа дозволяє користувачам швидко знаходити транспортні засоби, порівнювати пропозиції та розміщувати власні оголошення, забезпечуючи зручний пошук і широкий вибір.

Таблиця 1.4 – Портал оголошень з продажу авто RST.ua

Назва	RST.ua автобазар, оголошення про продаж авто в Україні.
Розробник	RST – незалежний український сайт-портал. Власна команда, яка оперує ресурсом.
Архітектура	Дошка оголошень (клієнт-сервер).
Мова реалізації	HTML/CSS/JavaScript.
Перелік функцій, характеристик	1) великий каталог оголошень (б/в та нові авто) з фільтрами по регіону, марці, моделі, ціні; 2) розміщення оголошення; 3) пошук за регіонами, категоріями, містами; 4) сторінки автосалонів/продавців, контактні дані, техпідтримка; 5) адаптивні сторінки та мобільна версія; 6) додаткові розділи: автоновини, правила публікації.
Переваги	1) велика база оголошень (масштабність і охоплення по регіонах); 2) зручні фільтри та можливість швидко порівнювати пропозиції.
Недоліки	1) менше офіційності та гарантії даних (різні приватні продавці) – ризики шахрайства та неточностей; 2) велика конкуренція за увагу оголошень (платні просуваються).

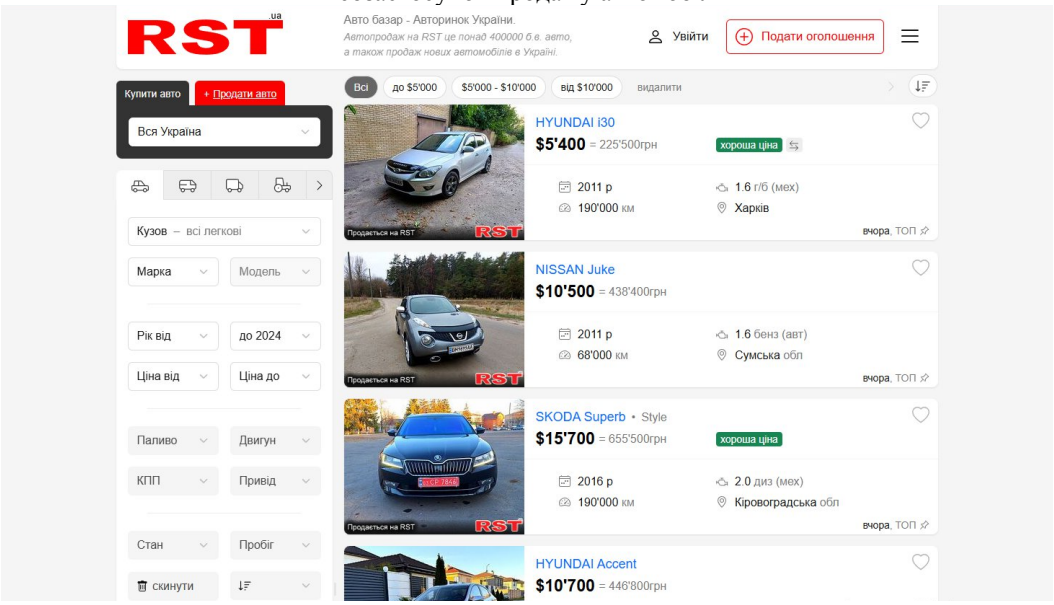

Рисунок 1.4 – Інтерфейс порталу оголошень RST.ua

Так виглядає вебзастосунок порталу оголошень RST.ua.

AUTO.RIA (табл. 1.5, рис. 1.5) – український онлайн-сервіс оголошень, орієнтований на купівлю та продаж транспортних засобів [7]. Платформа надає зручні інструменти для пошуку авто, перевірки інформації про транспорт та розміщення оголошень, забезпечуючи прозорість і широкий вибір пропозицій.

Таблиця 1.5 – Портал оголошень з продажу авто AUTO.RIA

Назва	AUTO.RIA – онлайн-портал продажу автомобілів.
Розробник	RIA.com.
Архітектура	Вебзастосунок (клієнт-сервер).
Мова реалізації	HTML/CSS/JavaScript, серверні технології (PHP).
Перелік функцій, характеристик	1) пошук автомобілів за параметрами (марка, модель, ціна, рік, пробіг); 2) розміщення приватних та дилерських оголошень; 3) перевірка авто (VIN-код, історія транспортного засобу); 4) порівняння автомобілів та аналітика цін; 5) вбудовані фільтри та рекомендації; 6) мобільний додаток для зручного доступу; 7) додаткові сервіси: автострахування, кредитування, новини авторинку.

Кінець таблиці 1.5

Переваги	1) велика база оголошень по всій Україні; 2) зручний та гнучкий пошук; 3) додаткові інструменти перевірки авто.
Недоліки	1) можливі недостовірні або застарілі оголошення; 2) відсутність гарантії та якості авто.

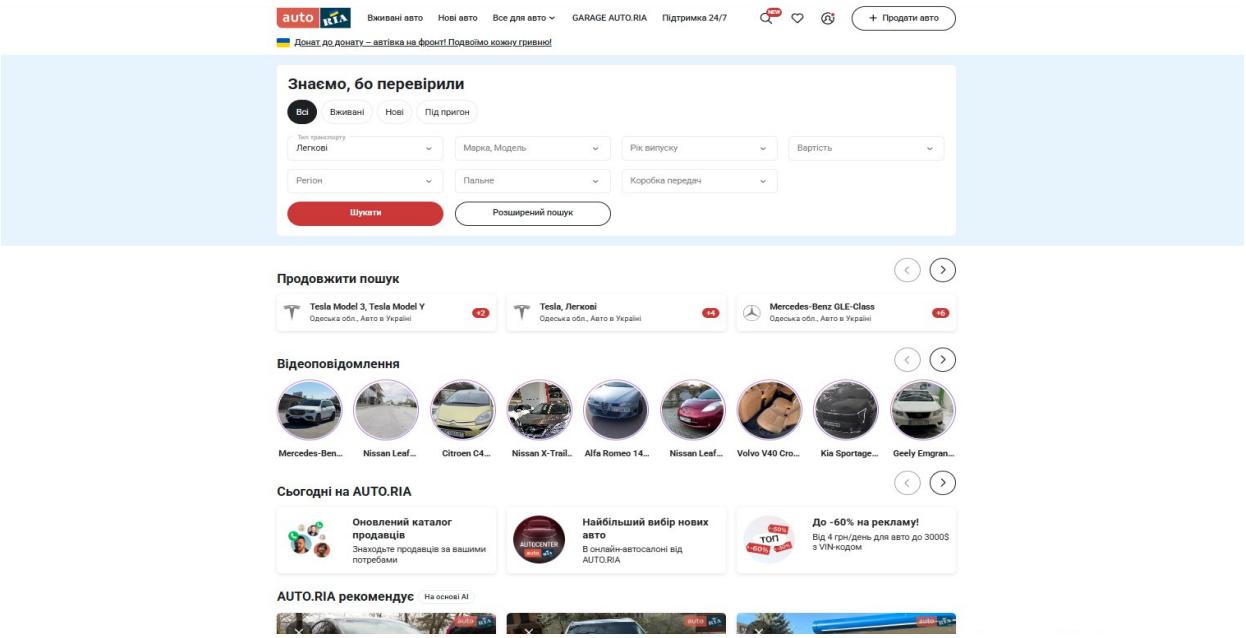

Рисунок 1.5 – Інтерфейс порталу оголошень AUTO.RIA

Даний вигляд має вебзастосунок порталу оголошень AUTO.RIA.

Аналіз існуючих рішень у сфері продажу автомобілів показав, що є суттєві відмінності між дилерськими сайтами та платформами оголошень. Встановлено, що офіційні дилери орієнтовані на продаж нових автомобілів конкретного бренду та забезпечують повний цикл обслуговування клієнта: від вибору авто до сервісного обслуговування. Їхніми основними перевагами є висока довіра, актуальність інформації, інтеграція з фінансовими сервісами та гарантійна підтримка. Також вони мають обмеження у вигляді вузької спеціалізації (лише один бренд) і відсутності можливості роботи з вторинним ринком.

Якщо розглядати портали оголошень можна визначити, що вони надають значно ширші можливості для користувачів, включаючи доступ до великої кількості пропозицій нових і вживаних автомобілів від приватних осіб і дилерів. Їхніми ключовими перевагами є масштабність, гнучкі інструменти пошуку та

додаткові сервіси (перевірка авто, аналітика, мобільний доступ). Однак такі платформи не мають високого рівня довіри та безпеки, тому що існує ризик недостовірної інформації або шахрайства.

1.3 Аналіз структурних і функціональних особливостей об’єкта дослідження

Структура системи

Вебзастосунок з продажу автомобілів реалізується на основі клієнт-серверної архітектури та складається з таких основних компонентів:

– клієнтська частина (інтерфейс користувача);
– серверна частина (бізнес-логіка);
– база даних (зберігання інформації).

Клієнтська частина реалізується за допомогою HTML, CSS та JavaScript і забезпечує взаємодію користувача із системою через веббраузер. Серверна частина відповідає за обробку запитів, виконання бізнес-логіки та взаємодію з базою даних. База даних зберігає інформацію про користувачів, автомобілі, оголошення та замовлення.

Учасники системи

Зазвичай у системі виділяють такі ролі користувачів:

– покупець – здійснює пошук автомобілів, переглядає інформацію, оформлює замовлення та записується на тест-драйв;
– продавець (автосалон) – розміщує оголошення про продаж автомобілів, редагує їх та аналізує статистику;
– адміністратор – редагує оголошення, керує користувачами та контентом, дивиться звіти та забезпечує стабільну роботу системи.

Функціональні можливості системи

Після проведення аналізу визначено основні функції, які характерні для подібних вебзастосунків:

1) для покупця:
   – реєстрація та авторизація;

- перегляд каталогу автомобілів;
- пошук та фільтрація;
- перегляд детальної інформації;
- додавання до обраного;
- порівняння автомобілів;
- оформлення замовлення на купівлю;
- вибір способу оплати;
- запис на тест-драйв.

2) для продавця:

- створення оголошень;
- редагування та видалення оголошень;
- завантаження фотографій автомобіля;
- перегляд статистики.

3) для адміністратора:

- управління користувачами;
- коригування оголошень;
- контроль роботи системи;
- формування звітів.

1.4 Постановка завдання

На основі проведеного аналізу предметної області, особливостей системи, та існуючих програмних рішень встановлено, що сучасні вебзастосунки для продажу автомобілів мають ряд обмежень.

Дилерські вебзастосунки забезпечують високий рівень надійності та достовірності інформації, проте орієнтовані лише на один бренд і не підтримують вторинний ринок. Портали оголошень, навпаки, мають широку базу пропозицій, але не гарантують достовірність даних та безпеку угод. У зв’язку з цим виникає необхідність розробки вебзастосунку, який поєднує переваги існуючих систем та усуває їх недоліки.

Тому потрібно розробити вебзастосунок для автоматизації процесу продажу автомобілів, що забезпечує багатокористувацький доступ до інформації про авто та надає зручні інструменти для їх пошуку, аналізу та придбання.

Висновки до розділу 1

У першому розділі проведено аналіз предметної області вебзастосунків для продажу автомобілів, що дозволило визначити основні особливості розвитку даної сфери. Встановлено, що інформаційні системи відіграють важливу роль у процесі купівлі та продажу транспортних засобів, забезпечуючи зручний доступ до інформації та спрощуючи взаємодію між користувачами. Визначення об’єкту та предмету дослідження дозволило визначити межі розроблюваної системи та основні напрямки її реалізації.

Під час аналізу існуючих програмних рішень встановлено, що дилерські вебзастосунки забезпечують надійність і достовірність інформації, але обмежені одним брендом. Портали оголошень, навпаки, надають широкий вибір автомобілів, проте мають ризики, пов’язані з достовірністю даних та безпекою угод.

На основі проведеного аналізу сформульовано задачу кваліфікаційної роботи, що полягає у розробці вебзастосунку для автоматизації процесу продажу автомобілів.

2 МОДЕЛЮВАННЯ ОБ'ЄКТУ ТА ПРЕДМЕТУ

2.1 Аналіз сучасного стану інструментарію реалізації вебзастосунків

Вебзастосунки продажу автомобілів працюють з великою кількістю оголошень, користувачів та запитів, що вимагає застосування сучасних архітектурних підходів. Розробка такого вебзастосунку потребує використання технологій, що забезпечують ефективну обробку даних, зручність користування, масштабованість та безпеку системи.

Розглядаючи сучасні наукові дослідження, можна визначити, що вебзастосунки електронної комерції повинні базуватися на масштабованих архітектурах, які дозволяють обробляти великі обсяги даних та забезпечують високу продуктивність системи. Одним із найбільш поширених підходів є клієнт-серверна архітектура (рис. 2.1), яка забезпечує розподіл функцій між інтерфейсом користувача та серверною частиною.

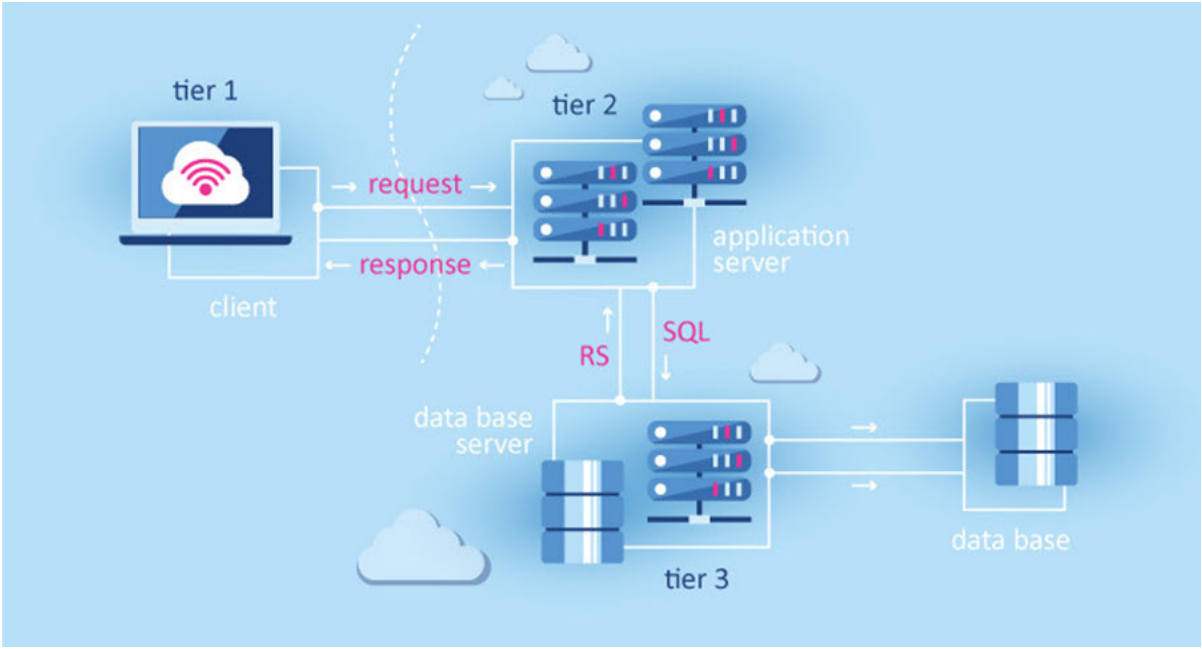

Рисунок 2.1 – Клієнт-серверна архітектура [8]

Також, у розглянутих наукових роботах досліджується використання мікросервісної архітектури та монолітної архітектури. Мікросервісна дозволяє розділити систему на незалежні компоненти, що можуть масштабуватися окремо. Монолітна пропонує простіший підхід та кращі результати для невеликих масштабів

системи [9, 10]. Мікросервісна архітектура підвищує гнучкість системи та спрощує її супровід, однак потребує більш складної реалізації, порівнюючи з монолітною.

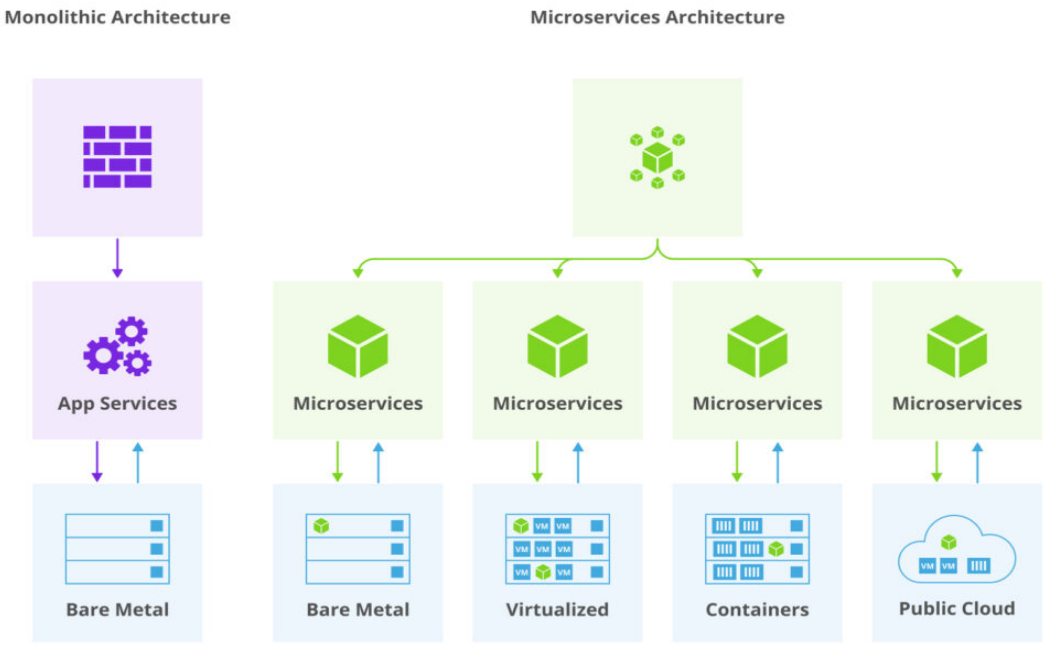

Рисунок 2.2 – Монолітна та мікросервісна архітектури [11]

Для створення користувацького інтерфейсу вебзастосунків використовуються технології HTML5, CSS3 та JavaScript (рис. 2.3), які формують структуру, стилізацію та інтерактивність вебсторінок [12, 13].

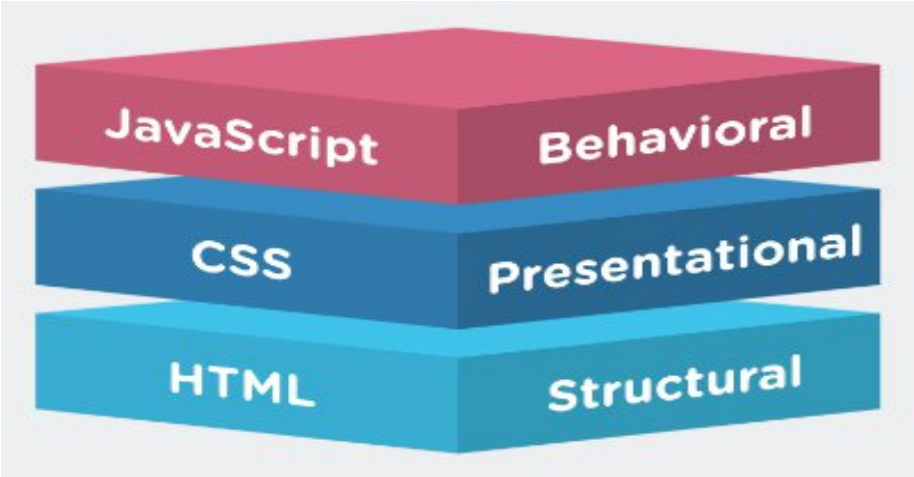

Рисунок 2.3 – Ролі HTML, CSS та JavaScript у формуванні вебінтерфейсу [14]

Серед сучасних фреймворків увагу приділяють Angular, React та Vue (табл. 2.1), які дозволяють реалізовувати односторінкові застосунки (SPA). Згідно з дослідженнями використання цих фреймворків дозволяє зменшити навантаження на

сервер, забезпечити швидке оновлення інтерфейсу без перезавантаження сторінки, підвищити продуктивність вебзастосунків та покращити взаємодію з користувачем [15].

Таблиця 2.1 – Angular, React та Vue

Характеристика	Angular	React	Vue
Підхід до розробки	На основі TypeScript	Все на JavaScript	На основі JavaScript та HTML
Коли обирати	Для розробки гібридних застосунків, або вебзастосунків	Для розробки SPA та мобільних застосунків для різних платформ	Для розробки розширених SPA
Використання	Створення великомасштабних, багатофункціональних застосунків, або застосунків корпоративного рівня	Створення сучасних вебзастосунків та застосунків для Android та iOS	Створення вебзастосунків та односторінкових застосунків
Зручність	Використання структурного фреймворку	Забезпечення гнучкого середовища розробки	Зосередженість на розділені завдань
Модель	На основі архітектури MVC (Model – View – Controller)	На основі Virtual DOM (Document object model)	На основі Virtual DOM (Document object model)
Написано на	TypeScript	JavaScript	JavaScript
Мова	Рекомендується використання TypeScript	Рекомендується використання JSX (JavaScript XML)	Шаблони HTML та JavaScript
Можливість повторного використання	Так	Ні, лише CSS	Так, HTML і CSS
Масштабованість	Модульна сруктура розробки	Компонентний підхід	Підхід на основі шаблонів

Серверна частина вебзастосунків може реалізовуватися з використанням різних технологій, таких як: Node.js, Java Spring, ASP.NET (табл. 2.2). Використання сучасних серверних платформ забезпечує високу продуктивність, безпеку та масштабованість систем [16].

Таблиця 2.2 – Node.js, Java Spring та ASP.NET

Характеристика	Node.js	Java Spring	ASP.NET
Мова програмування	JavaScript / TypeScript	Java	C#, F#
Архітектура	Однопотокова, асинхронна	Багатопотокова, класична обробка запитів	Багатопотокова
Продуктивність	Стабільно для великої кількості одночасних підключень	Висока надійність та стабільність	Дуже висока швидкість виконання, оптимізована під сервери Microsoft, висока продуктивність
Екосистема	Великий вибір модулів (npm), швидкий старт, слабке типування	Сувора типізація, корпоративні стандарти, велика кількість готових рішень	Тісна інтеграція з інструментами Microsoft (Visual Studio, Azure)
Використання	Стартапи, вебзастосунки в реальному часі (Real-time)	Великі корпоративні системи, банки, страхові компанії, промисловість	Корпоративні системи, інтеграція з екосистемою Windows, потужні вебзастосунки

Для збереження даних у вебзастосунках електронної комерції найчастіше використовуються реляційні бази даних, такі як: MySQL, PostgreSQL та Microsoft SQL Server (табл. 2.3). Вони забезпечують цілісність даних, підтримку транзакцій та ефективне виконання складних запитів [17].

Таблиця 2.3 – MySQL, PostgreSQL та Microsoft SQL Server

Характеристика	MySQL	PostgreSQL	SQL Server
Розробник	Oracle Corporation	Глобальна група розробників PostgreSQL	Microsoft
Ліцензія	З відкритим вихідним кодом (GPL), комерційна	З відкритим вихідним кодом (ліцензія PostgreSQL)	Власницька версія (з безкоштовною Express-версією)
Підтримка операційних систем	Кросплатформний	Кросплатформний	Windows, Linux (обмежена підтримка)
Відповідність ACID	Так (з рушієм InnoDB)	Повністю сумісний	Повністю сумісний
Відповідність стандартам	Помірна	Висока (близько до стандартів SQL)	Висока
Продуктивність	Швидка для робочих навантажень з великим обсягом читання даних	Краще для складних запитів та записів	Оптимізована для корпоративних навантажень
Підтримка JSON	Так	Розширена підтримка JSON та індексації	Так
Процедурна мова	SQL, обмежені збережені процедури	PL/pgSQL, PL/Python, PL/Perl, тощо	T-SQL
Підтримка	Комерційна підтримка від Oracle	Комерційна підтримка (EDB)	Сильна комерційна підтримка від Microsoft
Інструменти графічного інтерфейсу	MySQL Workbench, phpMyAdmin	pgAdmin, Dbeaver	SQL Server Management Studio
Безпека	Добра, SSL, дозволи користувачів	Сильна, захист на рівні рядків, SSL	Сильна, інтегрована з Windows AD

Кінець таблиці 2.3

Варіанти використання	Вебзастосунки, CMS	Аналітика, геопросторові застосунки, фінанси	Корпоративні застосунки, бізнес-аналітика, великомасштабні бази даних

Також важливим елементом сучасної розробки є використання систем контролю версій. Git (рис. 2.4) дозволяє розробляти проєкт командою, зберігає історію змін та забезпечує стабільність програмного продукту.

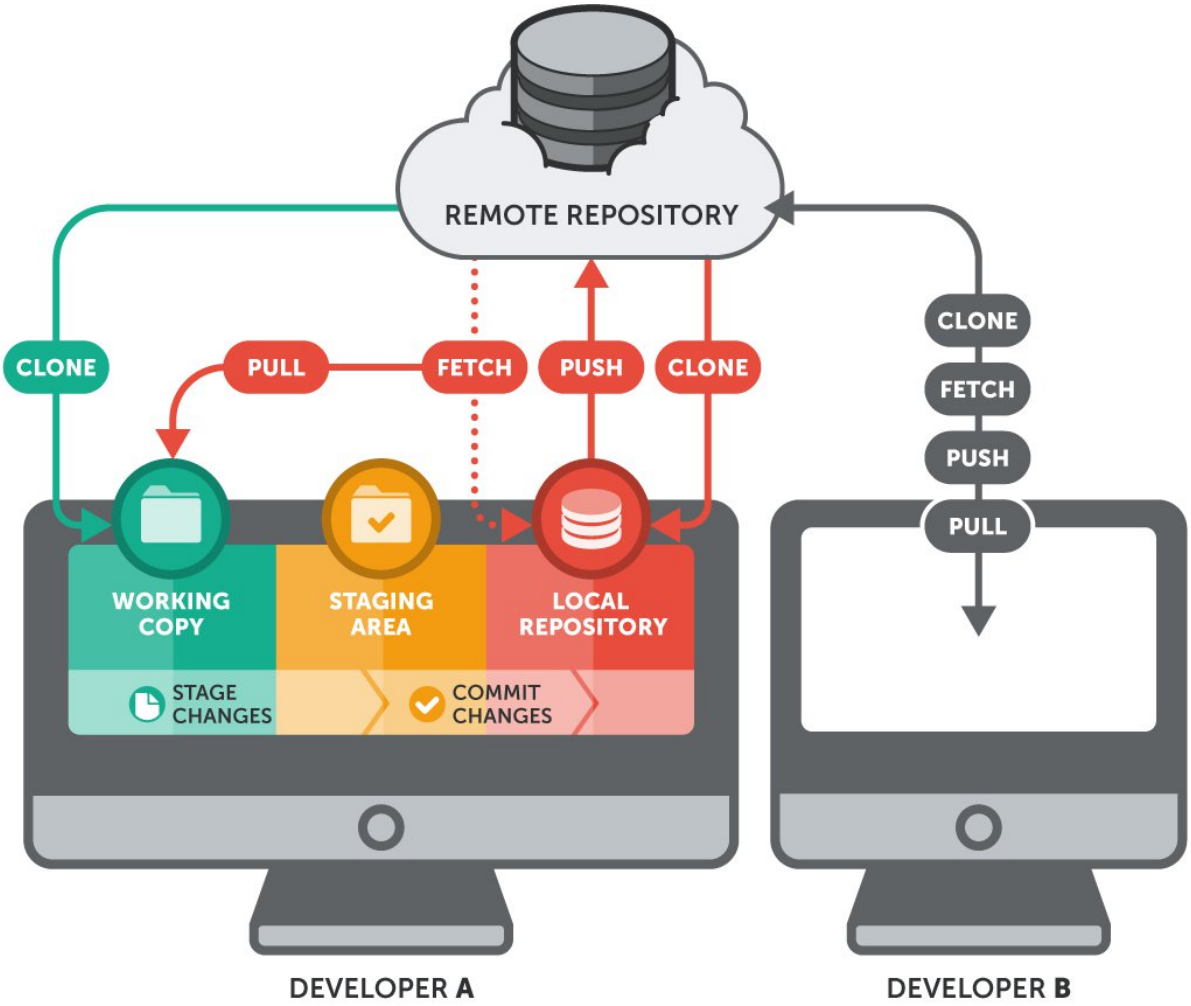

Рисунок 2.4 – Схема роботи системи контролю версій Git [18]

Проаналізувавши стан інструментарію можна сказати, що для розробки вебзастосунку продажу автомобілів доцільно використовувати клієнт-серверну архітектуру, сучасні серверні платформи та реляційні бази даних для ефективної реалізації функціональних можливостей системи.

2.2 Методи та моделі розробки вебзастосунку

Процес розробки вебзастосунку продажу автомобілів засновується на використанні сучасних методів програмної інженерії. Це дозволяє забезпечити якість, надійність та масштабованість програмного забезпечення.

Одним із найбільш ефективних підходів є ітеративно-інкрементна модель розробки. Вона передбачає створення системи поетапно з поступовим додаванням функціональних можливостей та покращенням вже існуючих. Такий підхід дозволяє враховувати зміни вимог та зменшити ризики розробки складних систем.

Також велика увага приділяється концепції web engineering. Вона передбачає системний підхід до розробки вебзастосунків і включає аналіз вимог, проєктування, реалізацію та тестування [19]. Використання цього підходу дозволяє підвищити якість програмного забезпечення та забезпечити відповідність потребам користувачів.

Розробка системи грунтується на об’єктно-орієнтованому підході, що дозволяє представити предметну область у вигляді сукупності взаємопов’язаних об’єктів, таких як: користувач, автомобіль, оголошення, заява на купівлю. Це забезпечує модульність системи, повторне використання коду та спрощує її подальший супровід [20].

При побудові архітектури системи можуть застосовуватися різні підходи. У сучасних вебзастосунках використовується клієнт-серверна архітектура, яка визначає взаємодію між клієнтською та серверною частинами системи. Разом цим внутрішня організація серверної частини може реалізовуватись за допомогою монолітної або мікросервісної архітектури. Для великих систем кращим є використання мікросервісного підходу, що дозволяє незалежно масштабувати компоненти системи та має такі переваги: покращена доступність, відмовостійкість та горизонтальна масштабованість, а також велика гнучкість розробки програмного забезпечення. Однак, для систем, що не мають тисяч одночасно працюючих користувачів та здатних до вертикального масштабування, перехід від монолітної архітектури до мікросервісної може не принести переваги, про які повідомляють

провідні світові компанії. Тому для невеликих проєктів ефективним є використання монолітної архітектури [9, 10]. Також пропонується використовувати гібридний метод масштабування, заснований на поєднанні двох підходів. Мікросервісний підхід для високонавантажених і потребуючих масштабування частин системи та монолітний підхід для ненавантажених частин системи [10].

Для моделювання предметної області та структури системи застосовуються стандартизовані нотації, такі як UML та ER-моделювання. Вони дозволяють візуалізувати структуру системи, взаємодію користувачів із системою, структуру даних та бізнес-процеси.

Для забезпечення якості програмного забезпечення застосовуються різні методи тестування:

– модульне тестування;
– інтеграційне тестування;
– функціональне тестування.

Таким чином, розглянуті сучасні методи та моделі розробки дозволяють створити ефективний, надійний та масштабований вебзастосунок продажу автомобілів.

2.3 Специфікація вимог до програмного забезпечення

1. Призначення та межі проєкту

1.1 Призначення системи

Програмне забезпечення призначене для створення вебзастосунку для забезпечення багатокористувацького доступу до товарів і послуг з продажу авто та взаємодії між продавцями і покупцями транспортних засобів.

1.2 Погодження, що ухвалені в програмній документації

Система розробляється як вебзастосунок (за допомогою сучасних фреймворків) із клієнт-серверною архітектурою (реляційна база даних). Доступ до системи здійснюється через веббраузер.

1.3 Межі проєкту ПЗ

– реалізація вебверсії з можливістю адаптації під різні пристрої;

– використання сторонніх API;

– підтримка клієнтського та адміністративного інтерфейсів через єдину платформу.

2. Загальний опис

2.1 Сфера застосування

Система використовується компанією, яка надає можливість автосалонам та приватним продавцям розміщувати оголошення про продаж автомобілів.

2.2 Характеристики користувачів

– покупець – здійснює пошук автомобілів, переглядає інформацію, оформлює замовлення (якщо зареєстрований) та записується на тест-драйв (якщо зареєстрований);

– продавець (автосалон) – розміщує оголошення про продаж автомобілів, редагує їх та аналізує статистику;

– адміністратор – керує користувачами та контентом, дивиться звіти та забезпечує стабільну роботу системи.

2.3 Загальна структура системи

Система складається з:

– адаптивного вебінтерфейсу користувача;

– серверної логіки;

– реляційної бази даних.

2.4 Загальні обмеження

– доступ через Інтернет;

– підтримка сучасних браузерів;

– кількість одночасних користувачів яку дозволяють можливості сервера.

3. Функції системи

3.1 Реєстрація користувача

Функція дозволяє створити обліковий запис. Вхідна інформація – електронна пошта, пароль та ім’я користувача. Вихідна інформація – вхід/створення облікового

запису. Функціональні вимоги: перевірка унікальності email; збереження даних користувача.

3.2 Перегляд каталогу автомобілів

Функція відображає список автомобілів. Вхідна інформація – параметри фільтрації (марка, ціна, рік). Вихідна інформація – потрібний список авто. Функціональні вимоги: відображення списку авто; сортування результатів.

3.3 Перегляд інформації про автомобіль

Функція відображає характеристики автомобіля. Вхідна інформація – ID автомобіля. Вихідна інформація – детальна інформація про обране авто. Функціональні вимоги: відображення фото; опис характеристик; відображення ціни.

3.4 Оформлення заяви на покупку

Функція створює заяву на купівлю. Вхідна інформація – обраний автомобіль; контактні дані користувача. Вихідна інформація – повідомлення (SMS або електроний лист) про створення та збереження заяви. Функціональні вимоги: створення заяви; збереження до бази даних.

3.5 Адміністрування каталогу

Функція додавання, редагування та видалення автомобілів. Вхідна інформація – дані автомобіля. Вихідна інформація – оновлення списку авто. Функціональні вимоги: створення оголошень; редагування характеристик; видалення оголошення про авто.

3.6 Порівняння автомобілів

Функція порівняння автомобілів. Вхідна інформація – ID автомобіля, дані автомобіля. Вихідна інформація – результати порівняння у вигляді таблиці. Функціональні вимоги: додавання обраних авто до таблиці порівнянь; відображення інформації про обрані авто.

4. Вимоги до інформаційного забезпечення

4.1 Джерела і зміст вхідної інформації

– дані користувачів (ім’я, прізвище, контактна інформація);
– дані автомобілів від офіційних дилерів.

4.2 Нормативно-довідкова інформація

– список марок автомобілів;

– список моделей;

– типи автомобілів.

4.3 Організація збереження інформації

Інформація зберігається у реляційній базі даних. Регулярно робиться резервне копіювання.

5. Вимоги до технічного забезпечення

– сервер з підтримкою вебсерверного ПЗ;

– клієнтський пристрій із веббраузером;

– доступ до мережі Інтернет.

6. Вимоги до програмного забезпечення

6.1 Архітектура системи

Клієнт-серверна архітектура.

6.2 Системне програмне забезпечення

Вебзастосунок реалізовується за допомогою фреймворку Angular. В якості бази даних системи виступає Microsoft SQL Server.

6.3 Мережеве ПЗ

Використання протоколу HTTPS.

6.4 ПЗ ведення бази даних

Система керування базами даних Microsoft SQL Server.

6.5 Мови та технології

Frontend: HTML, CSS, JavaScript, Angular, TypeScript. Backend: ASP.NET. База даних: система керування базами даних Microsoft SQL Server.

7. Вимоги до зовнішніх інтерфейсів

7.1 Інтерфейс користувача

Графічний вебінтерфейс із адаптивним дизайном.

7.2 Апаратний інтерфейс

Сучасний пристрій з доступом до веббраузера через Інтернет.

7.3 Програмний інтерфейс

REST API для обміну даними.

7.4 Комунікаційний протокол

HTTPS.

8. Властивості програмного забезпечення

8.1 Доступність

Система доступна цілодобово (24/7), за винятком часу проведення технічних робіт або аварійних ситуацій на стороні серверного обладнання чи хостинг-провайдера. Вебзастосунок має коректно працювати на різних типах пристроїв (ПК, ноутбуки, планшети, смартфони) та забезпечувати доступ до інформації без необхідності встановлення додаткового програмного забезпечення.

8.2 Супроводжуваність

Програмне забезпечення повинно бути зручним для супроводу та оновлення.

8.3 Переносимість

Програмне забезпечення повинно бути переносимим та мати можливість розгортання на іншому сервері або хостингу без суттєвих змін у коді.

8.4 Продуктивність

Програмне забезпечення повинно забезпечувати стабільну та швидку роботу при нормальному навантаженні.

8.5 Надійність

Система повинна працювати стабільно та коректно протягом тривалого часу. Забезпечення резервного копіювання.

8.6 Безпека

– шифрування даних;
– автентифікація користувачів;
– захист від SQL-ін’єкцій.

9. Інші вимоги

Система повинна:

– мати можливість подальшого розширення функціоналу;

– відповідати вимогам чинних стандартів та рекомендацій щодо розробки вебзастосунків;

– мати адаптивний інтерфейс, який буде коректно відображатися на різних розширеннях екранів.

Висновки до розділу 2

У другому розділі проведено аналіз сучасного стану інструментарію, методів та моделей, що використовуються для розробки вебзастосунків електронної комерції. Визначено основні технологічні підходи, які дозволяють забезпечити ефективну реалізацію системи продажу автомобілів.

Розглянуто сучасні методи розробки програмного забезпечення, такі як ітеративно-інкрементний підхід та об’єктно-орієнтоване програмування, що забезпечують гнучкість та масштабованість системи.

Сформовано специфікацію вимог до програмного забезпечення, яка визначає функціональні та нефункціональні характеристики системи, а також основні вимоги до її реалізації і можливості підтримки у майбутньому.

Отримані результати проведеного аналізу є основою для подальшого проєктування архітектури системи, розробки UML-діаграм та реалізації програмного забезпечення.

3 ПРОЄКТУВАННЯ ТА КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ

Проєктування програмного забезпечення є одним з головних етапів розробки IT-продукту, що визначає його структуру, функціональність, здатність до масштабування, та безперервний розвиток в майбутньому. Воно передбачає розробку чіткого плану дій, вибір представлення даних, стилів та шаблонів архітектури і визначення стратегій вирішення завдань.

Проєктування програмного забезпечення робиться для таких цілей:

– дозволяє зрозуміти, який використовувати стек технологій для розробки та які методи розробки використовувати;

– дає можливість попередньо оцінити вартість розробки та терміни реалізації проєкту;

– мінімізує ризики в ході розробки та виключає непотрібні дії;

– спрощує узгодження ходу розробки з клієнтом та дає чітке уявлення про можливості майбутнього продукту;

– дозволяє зрозуміти як працюватиме IT-продукт, як його можна масштабувати та покращувати його функціонал;

– зменшує розбіжності у баченні IT-продукту між клієнтом та виконавцем.

Проєктування програмного продукту потрібне для того, щоб створити чітке технічне завдання, побачити можливості та функціонал програми до початку розробки та зрозуміти алгоритм дій.

Проєктування вебзастосунку виконується з використанням сучасних підходів та методів моделювання програмного забезпечення. У процесі проєктування застосовуються функціональні, структурні, інформаційні та поведінкові моделі, що дозволяють описати роботу системи, взаємодію між її компонентами та основні сценарії використання.

Для представлення структури та логіки роботи системи використовуються UML-діаграми, ER-моделі, алгоритмічні схеми та інші засоби моделювання. Їх використання дозволяє змоделювати взаємодію користувачів із системою,

структуру даних, взаємозв’язки між сутностями та логіку виконання основних функцій вебзастосунку.

Проєктування програмного забезпечення охоплює розробку структури бази даних, побудову архітектури системи, визначення взаємозв’язків між модулями та проєктування основних алгоритмів роботи застосунку. Процес моделювання дає можливість оцінити ефективність роботи системи, виявити потенційні проблеми ще до етапу реалізації та сформувати основу для подальшої розробки програмного забезпечення.

3.1 Архітектура та стек технологій

Архітектура сучасних вебзастосунків електронної комерції будується за принципом клієнт-серверної взаємодії, що забезпечує розділення функцій між клієнтською частиною, серверною логікою та системою збереження даних. Такий підхід дозволяє забезпечити масштабованість, зручність супроводу та ефективну обробку запитів користувачів.

Основою взаємодії між компонентами системи є REST API. Через нього клієнтська частина надсилає HTTP-запити до серверної частини та отримує відповіді у форматі JSON. Це дозволяє забезпечити незалежність frontend та backend частин системи, спрощує обмін даними та полегшує подальше масштабування застосунку.

Frontend (клієнтська частина) відповідає за взаємодію користувача із системою, формування інтерфейсу, обробку подій та відображення інформації. Для реалізації клієнтської частини використовуються сучасні вебтехнології:

– HTML5 – для створення структури вебсторінок та опису елементів інтерфейсу;

– CSS3 – для оформлення зовнішнього вигляду вебзастосунку, створення адаптивного дизайну та реалізації анімацій;

– JavaScript – використовується для підтримки роботи окремих бібліотек та взаємодії браузера з вебсторінкою;

– Angular – фреймворк для створення односторінкового застосунку (SPA). Він дозволяє реалізувати компонентну архітектуру, маршрутизацію, двосторонню взаємодію з даними та оновлення інтерфейсу без перезавантаження сторінки [21];

– TypeScript – як основна мова програмування клієнтської частини, що забезпечує типізацію, покращує читабельність коду та спрощує супровід великих проєктів.

Також важливо приділити увагу адаптивності інтерфейсу клієнтської частини, що дозволяє коректно відображати вебзастосунок на різних пристроях.

Backend (серверна частина) реалізується за допомогою платформи ASP.NET [22]. Вона забезпечує обробку запитів користувачів, виконання бізнес-логіки, роботу з базою даних та реалізацію механізмів безпеки системи. Серверна частина відповідає за:

– авторизацію та автентифікацію користувачів;
– обробку інформації про автомобілі;
– створення та редагування оголошень;
– оформлення заявок на купівлю/тест-драйв;
– взаємодію з базою даних;
– формування відповідей REST API.

Для збереження інформації використовується система керування базами даних Microsoft SQL Server. Вона забезпечує надійне збереження інформації, підтримку транзакцій та цілісність даних. У базі даних зберігаються:

– дані користувачів;
– інформація про автомобілі;
– заявки на купівлю/тест-драйв;
– службова інформація системи.

Взаємодія між компонентами системи здійснюється наступним чином: користувач через вебінтерфейс виконує певну дію (наприклад, перегляд автомобілів або створення заявки), після чого Angular-застосунок надсилає HTTP-запит до REST API серверної частини. Сервер обробляє запит, взаємодіє з базою

даних Microsoft SQL Server та повертає клієнту відповідь у форматі JSON. Отримані дані оброблюються клієнтською частиною та відображаються користувачу без перезавантаження сторінки.

Для забезпечення безпеки застосунку використовуються механізми:

– автентифікації та авторизації користувачів;
– валідації введених даних;
– захисту від SQL-ін’єкцій;
– використання протоколу HTTPS для шифрування переданих даних.

Обраний стек технологій забезпечує високу швидкість роботи вебзастосунку, спрощує подальший супровід програмного забезпечення та дозволяє реалізувати сучасний адаптивний інтерфейс користувача.

Перевагами обраної архітектури є:

– модульність та зручність супроводу системи;
– незалежність клієнтської та серверної частин;
– можливість масштабування системи;
– швидка взаємодія користувача із застосунком;
– підтримка роботи на різних пристроях та браузерах;
– високий рівень безпеки даних.

Таким чином, обрана архітектура та компоненти вебзастосунку забезпечують швидку реалізацію функціональних можливостей системи продажу автомобілів та створюють основу для подальшого розвитку програмного забезпечення.

3.2 Функціональне моделювання

Функціональне моделювання – це важливий етап проєктування, що дозволяє графічно представити основні бізнес-процеси, взаємодію користувачів з системою, роботу програмних компонентів та визначити дані, які будуть оброблятися в межах вебзастосунку. Для цього використовується методологія IDEF0, яка є стандартом у галузі структурного аналізу та моделювання інформаційних систем.

IDEF0 – це визначення інтеграції для моделювання процесів, методологія яка використовується для моделювання бізнесу та їхніх процесів, щоб їх можна було

зрозуміти та вдосконалити. Контекстна діаграма IDEF0 – це найвищий рівень будь-якого структурно-функціонального моделювання. Вона зображає систему у вигляді єдиної головної функції та показує її взаємодію із зовнішнім середовищем (рис. 3.1).

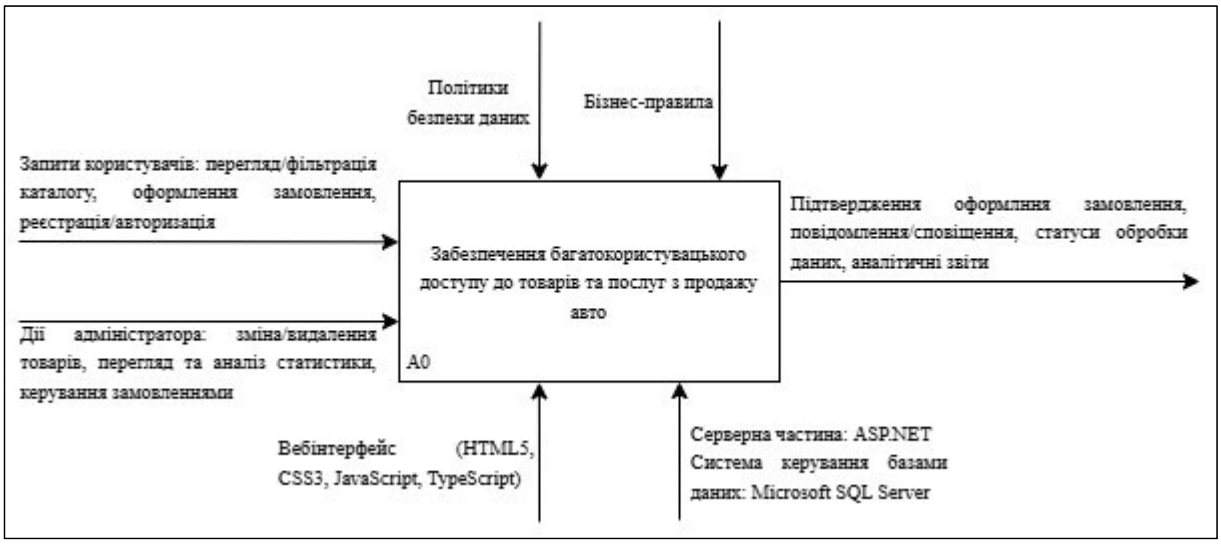

Рисунок 3.1 – Контекстна діаграма IDEF0

Контекстна діаграма IDEF0 складається з:

– функціонального блоку який позначає головну функцію всієї системи;

– входів (стрілки, які входять зліва): матеріали, що перетворюються системою на вихідний продукт;

– керування (стрілки, які входять зверху): правила, стандарти, закони або інструкції, які спрямовують чи обмежують процес;

– механізмів (стрілки, які входять знизу): ресурси, обладнання або персонал, які виконують роботу;

– виходів (стрілки, які виходять справа): кінцевий результат процесу (продукт або послуга).

Після побудови контекстної діаграми потрібно єдиний блок основної функції системи розбити на 3 основних підпроцеси, щоб деталізувати внутрішню роботу системи:

– A1 – операції пов’язані з каталогом товарів (оформлення замовлень, фільтрація, порівняння);

– A2 – керування користувачами та доступом;

– A3 – аналітика та адміністрування.

Кожен з цих підпроцесів має свої вхідні матеріали, керування, механізми реалізації та вихідні результати (рис. 3.2).

Рисунок 3.2 – Діаграма декомпозиції

Діаграма декомпозиції деталізує головну функцію застосунку який розроблюється. Вона демонструє поділену систему на три основні підпроцеси: операції пов’язані з каталогом товарів (A1), керування користувачами та доступом (A2), аналітика та адміністрування (A3). Такий підхід дає можливість детально представити головні бізнес-процеси системи та визначити для кожного з них вхідні матеріали, керування, механізми реалізації та вихідні результати. Також цей підхід дає змогу показати взаємозв’язок між компонентами в межах інформаційного середовища, що спрощує процес проєктування, тестування та подальшого розвитку програмного забезпечення.

3.3 Використання системи

Варіанти використання розробляються для чіткого визначення функціональності системи. Вони перетворюють вимоги на зрозумілі сценарії, що

забезпечують зв’язок між бізнес-проєктом та розробниками. Варіанти використання створюються для:

– формування вимог: визначення меж проєкту та того, що саме повинна робити система;

– тестування: створення чітких критеріїв для перевірки готовності продукту;

– планування розробки: розподіл великого проєкту на логічні кроки для оцінки обсягу роботи та пріоритетності.

1) Коротка форма.

Реєстрація або авторизація користувача: користувач відкриває вебзастосунок та натискає кнопку “Login”. Система відкриває модальне вікно, де користувач обирає що він хоче зробити: зареєструватися (якщо це перший вхід) або увійти в обліковий запис. Після цього відкривається форма при заповнені якої користувач входить в свій обліковий запис.

Перегляд каталогу автомобілів: після того як користувач відкрив вебзастосунок він може перейти до перегляду каталогу автомобілів. Після його відкриття система відображає повний каталог автомобілів до якого можуть застосовуватися фільтри або пошук за назвою авто. Після застосування фільтрації каталог товарів змінюється і система відображає перелік автомобілів, який відповідає застосованим фільтрам.

Перегляд інформації про певний автомобіль: при перегляді каталогу автомобілів користувач може натиснути на картку обраного авто і система відкриє сторінку з детальною інформацією про нього. Після цього користувач може ознайомитися з фотографіями обраного авто та усією інформацією стосовно цього товару.

Оформлення замовлення: під час перегляду детальної інформації про певне авто можна натиснути на кнопку “Buy”, після чого система відкриє сторінку з формою для оформлення замовлення. Під час заповнення форми відбувається перевірка введених даних. При неправильних введених даних користувачу виведеться повідомлення про неправильні введені дані. Якщо вся введена

інформація правильна, створюється запис замовлення у базі даних, а користувачу висвічується сповіщення про збережене замовлення.

Технічна підтримка користувачів та зворотній зв'язок: якщо під час користування вебзастосунком у користувача з'являється проблема у роботі вебзастосунку, або потрібна консультація, то користувач може звернутися до служби підтримки з тим питанням яке потрібно вирішити.

2) Поверхнева форма.

Реєстрація або авторизація користувача

Головний сценарій: користувач відкриває вебзастосунок та натискає кнопку "Login". Система відкриває модальне вікно, де користувач обирає що він хоче зробити: зареєструватися (якщо це перший вхід) або увійти в обліковий запис. Після цього відкривається форма, користувач вводить дані і натискає кнопку підтвердження. Після цього система перевіряє, чи існує в базі даних обліковий запис цього користувача. Якщо облікового запису немає, то система створює його. Після цього користувач отримує підтвердження успішного входу в свій обліковий запис.

Альтернативні сценарії:

– користувач не натискає кнопку підтвердження для завершення входу в свій обліковий запис. Система висвічує сповіщення про скасування входу;

– користувач вводить неправильні дані. Система повідомляє про невідповідність даних;

– користувач намагається увійти до облікового запису немаючи його. Система повідомляє про відсутність такого облікового запису і пропонує реєстрацію;

– під час входу в обліковий запис відбувається системний, або технічний збій. Система повідомляє про збій і пропонує спробувати пізніше.

Оформлення замовлення

Головний сценарій: користувач відкриває вебзастосунок та входить до свого облікового запису. Після цього переходить до каталогу автомобілів і відкриває сторінку детальної інформації про обране авто. Під час перегляду детальної

інформації натискає на кнопку “Buy”, після чого система відкриє сторінку з формою для оформлення замовлення. Під час заповнення форми відбувається перевірка введених даних. При неправильних введених даних користувачу виведеться повідомлення про неправильні введені дані. Якщо вся введена інформація правильна, створюється запис замовлення у базі даних, а користувачу висвічується сповіщення про збережене замовлення.

Альтернативні сценарії:

– користувач намагається оформити замовлення на авто, але його немає в наявності. Система повідомляє, що дане авто недоступне;

– користувач вводить неправильні дані при оформленні замовлення. Система повідомляє про невідповідність даних;

– під час заповнення форми відбувається збій. Система пропонує спробувати оформити замовлення пізніше;

– користувач не завершує заповнення форми для оформлення замовлення. Система зберігає замовлення у «відкладених».

Отримання сповіщень

Головний сценарій: при реєстрації у вебзастосунку користувач починає отримувати на електронну пошту (email) сповіщення про замовлення, або зміни інформації на сторінці новин.

Альтернативні сценарії:

– користувач відмовляється від отримання повідомлень. Система більше не надсилає повідомлення на вказаний email;

– відбувається технічний збій. Система проводить повторну спробу надіслати повідомлення.

3) Повна форма.

Таблиця 3.1 – Сценарій «Реєстрація/авторизація та перегляд каталогу товарів»

Usecase sections	Comment
Use Case Name	Реєстрація/авторизація та перегляд каталогу товарів
Scope	Вебзастосунок продажу автомобілів
Level	User-goal

Кінець таблиці 3.1

Primary Actor	Покупець
Stakeholders and Interests	Користувач хоче переглянути каталог та обрати потрібне авто. Адміністратор зацікавлений в реєстрації нових користувачів та клієнтів і в правильній роботі системи.
Preconditions	1) користувач має доступ до інтернету; 2) браузер підтримує відкриття вебзастосунку.
Success Guarantee	Користувач успішно реєструється/авторизується в своєму обліковому записі та переходить до перегляду каталогу авто.
Main Success Scenario	1) користувач відкриває вебзастосунок; 2) відкриває форму авторизації; 3) після введення даних підтверджує вхід в обліковий запис; 4) переходить до сторінки каталогу авто; 5) застосовує фільтри; 6) знаходить потрібне авто; 7) переглядає детальну інформацію про обране авто.
Extensions	1) вебзастосунок тимчасово недоступний. Система повідомляє про технічні роботи; 2) користувач не зміг увійти в свій обліковий запис. Система пропонує спробувати пізніше.
Special Requirements	1) стабільне підключення до інтернету; 2) наявність браузера який підтримує вебзастосунок.
Technology and Data Variations List	1) версія браузера яка підтримує роботу вебзастосунку.
Frequency of Occurrence	100%
Miscellaneous	Можливість перегляду каталогу авто без авторизації.

Таблиця 3.2 – Сценарій «Оформлення замовлення»

Usecase sections	**Comment**
Use Case Name	Оформлення замовлення
Scope	Вебзастосунок продажу автомобілів
Level	User-goal

Кінець таблиці 3.2

Primary Actor	Покупець
Stakeholders and Interests	Користувач зацікавлений у створенні замовлення на купівлю авто. Продавець зацікавлений у продажу авто. Адміністратор зацікавлений в правильній роботі системи.
Preconditions	1) користувач має доступ до інтернету; 2) браузер підтримує відкриття вебзастосунку.
Success Guarantee	Користувач успішно оформлює замовлення на купівлю авто.
Main Success Scenario	1) користувач відкриває вебзастосунок; 2) входить до свого облікового запису; 3) переходить до сторінки каталогу авто; 4) застосовує фільтри; 5) знаходить потрібне авто; 6) переходить до заповнення форми на оформлення замовлення; 7) підтверджує замовлення; 8) отримує відповідне сповіщення.
Extensions	1) вебзастосунок тимчасово недоступний. Система повідомляє про технічні роботи; 2) користувач не зміг увійти в свій обліковий запис. Система пропонує спробувати пізніше.
Special Requirements	1) стабільне підключення до інтернету; 2) наявність браузера який підтримує вебзастосунок; 3) користувач увійшов до облікового запису.
Technology and Data Variations List	1) версія браузера яка підтримує роботу вебзастосунку.
Frequency of Occurrence	60%

Таблиця 3.3 – Сценарій «Додавання нового авто»

Usecase sections	**Comment**
Use Case Name	Додавання нового авто
Scope	Вебзастосунок продажу автомобілів
Level	User-goal
Primary Actor	Продавець

Кінець таблиці 3.3

Stakeholders and Interests	Продавець зацікавлений у додаванні нового товару до каталогу. Адміністратор зацікавлений в правильній роботі системи. Користувач зацікавлений у можливості перегляду нових товарів.
Preconditions	1) продавець має доступ до інтернету; 2) браузер підтримує відкриття вебзастосунку; 3) продавець увійшов до свого облікового запису.
Success Guarantee	Продавець успішно додає товар до каталогу. Новий товар з'являється в каталозі і доступний для перегляду.
Main Success Scenario	1) продавець відкриває вебзастосунок; 2) входить до свого облікового запису; 3) переходить до сторінки каталогу авто; 4) натискає на кнопку додати авто; 5) заповнює форму для додавання нового авто; 6) підтверджує додавання; 7) отримує відповідне сповіщення.
Extensions	1) вебзастосунок тимчасово недоступний. Система повідомляє про технічні роботи; 2) продавець ввів некоректні дані. Система повідомляє про неправильні дані.
Special Requirements	1) стабільне підключення до інтернету; 2) наявність браузера який підтримує вебзастосунок; 3) перевірка інформації яку вносять в форму для додавання авто.
Technology and Data Variations List	1) версія браузера яка підтримує роботу вебзастосунку; 2) підтримка додавання фото через браузер.
Frequency of Occurrence	35%

Таблиця 3.4 – Сценарій «Керування товарами»

Usecase sections	Comment
Use Case Name	Керування товарами
Scope	Вебзастосунок продажу автомобілів
Level	User-goal
Primary Actor	Адміністратор
Stakeholders and Interests	Адміністратор зацікавлений в забезпеченні дотримання політик компанії та правильній роботі системи. Користувач зацікавлений у можливості перегляду товарів. Продавець зацікавлений в коректному відображені інформації про товар.
Preconditions	1) адміністратор має доступ до інтернету; 2) браузер підтримує відкриття вебзастосунку; 3) адміністратор увійшов до свого облікового запису з правами доступу до панелі управління.
Success Guarantee	Адміністратор увійшов до свого облікового запису, перевірив коректність відображення інформації, видалив товари які не відповідають тематиці вебзастосунку.
Main Success Scenario	1) адміністратор відкриває вебзастосунок; 2) входить до свого облікового запису; 3) переходить до сторінки каталогу авто; 4) перевіряє каталог на наявність товару який не відповідає тематиці вебзастосунку; 5) видаляє відповідні товари; 6) оновлює каталог; 7) перевіряє справність роботи вебзастосунку.
Extensions	1) адміністратор випадково видалив не той товар. Система має змогу відновити видалений товар; 2) адміністратор виявляє несправності в роботі вебзастосунку. Зв’язується з техпідтримкою і виправляють несправності.
Special Requirements	1) стабільне підключення до інтернету; 2) наявність браузера який підтримує вебзастосунок; 3) ведення обліку видалених/змінених товарів.

Кінець таблиці 3.4

Technology and Data Variations List	1) версія браузера яка підтримує роботу вебзастосунку; 2) логування видалених товарів.
Frequency of Occurrence	10%

Варіанти використання системи демонструють взаємодію користувачів вебзастосунку (клієнт, продавець, адміністратор) із основними функціями системи, такими як: реєстрація/авторизація, перегляд каталогу авто, перегляд детальної інформації про обране авто, порівняння характеристик обраних авто, оформлення замовлень та керування товарами.

На основі даних описаних в таблицях варіантів використання побудовано діаграму варіантів використання вебзастосунку (рис. 3.3) [23].

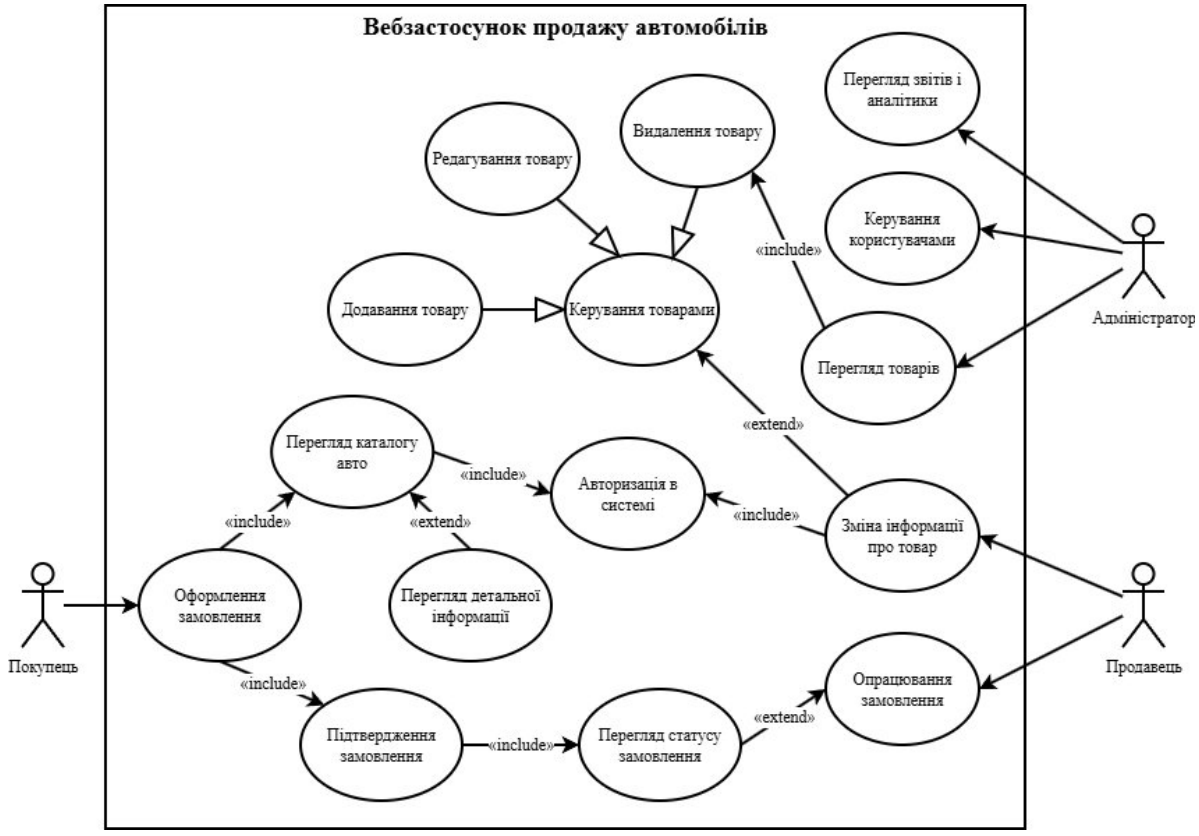

Рисунок 3.3 – Діаграма варіантів використання

Діаграма варіантів використання вебзастосунку продажу автомобілів містить трьох головних акторів: покупця, продавця та адміністратора. Кожен із них має різні права доступу до функцій системи.

Покупець може переглядати каталог товарів та оформлювати замовлення. Під час перегляду каталогу товарів покупець має можливість застосувати фільтрацію до каталогу для точного пошуку певного товару та перегляду їхніх детальних характеристик. Оформлення замовлення включає введення певної інформації про покупця, для ідентифікації замовлення та ведення звітності.

Продавець має змогу керувати товаром в каталозі. Його основні функції це додавати новий товар до каталогу, редагувати інформацію про товар, або видаляти його з каталогу. Це показує, що всі функції до яких продавець має доступ, використовуються для керування товарами в каталозі, що спрощує роботу з даними у системі.

Адміністратор відповідає за ключові функції роботи системи. Він перевіряє справність роботи вебзастосунку, має можливість керування користувачами та товарами в каталозі. Процес керування товарами включає в себе перевірку товарів та видалення тих, які не відповідають тематиці вебзастосунку. Також адміністратор володіє доступом до перегляду звітів та аналітичних даних, які допомагають оцінити ефективність роботи системи та вчасно виявити порушення або несправності.

Взаємозв’язки між діями у варіантах використання представлені за допомогою include, extend і generalization. Наприклад для продавця функція «Зміна інформації про товар» включає авторизацію в системі для підтвердження його прав доступу до зміни товарів. Разом з цим «Зміна інформації про товар» розширюється до загальної функції «Керування товарами», що в свою чергу використовує узагальнення окремих дії з інформацією про товари.

3.4 Графічне представлення роботи програмного забезпечення

Для графічного представлення роботи системи використовуються UML-діаграми – це стандартизовані графічні схеми, які використовуються для візуалізації та проєктування програмних систем, процесів і структур даних. Вони дозволяють у повному обсязі візуалізувати функціональність, структуру, логіку та взаємодію компонентів програмного забезпечення що розроблюється.

Діаграми послідовності – це поведінкові діаграми, які демонструють як різні об’єкти і компоненти вебзастосунку взаємодіють між собою для виконання певних завдань [24]. На цих діаграмах зображаються актори, основні об’єкти та компоненти системи. Між ними визначаються повідомлення, що демонструють виклики методів або передачу даних (рис. 3.4 – 3.6).

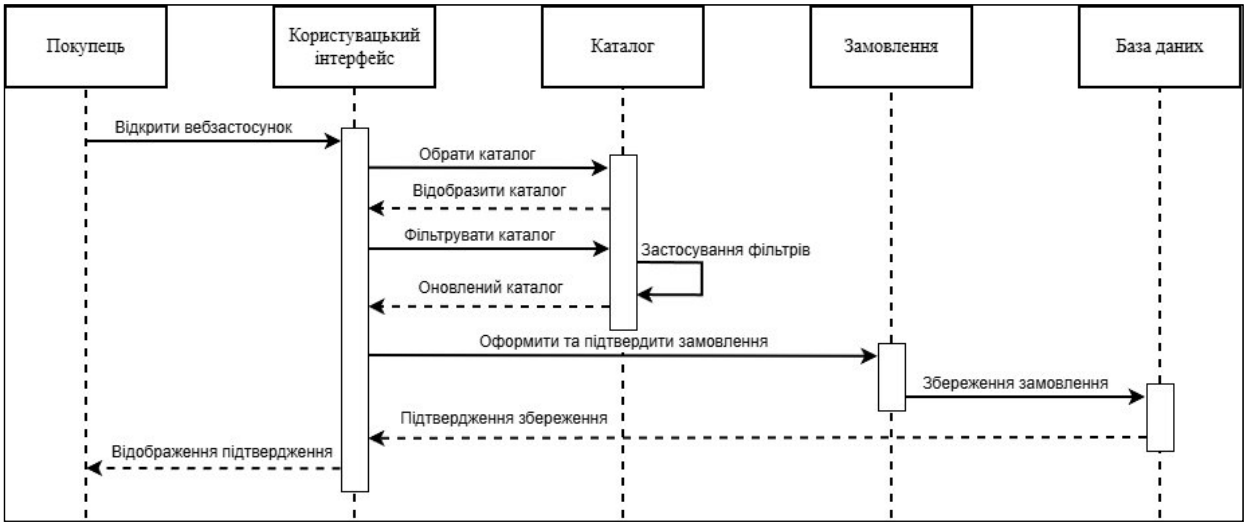

Рисунок 3.4 – Перегляд каталогу та оформлення замовлення

На діаграмі зображено послідовність дій під час перегляду каталогу товарів:

– покупець відкриває каталог у вебзастосунку;
– система відображає каталог товарів з можливістю фільтрації;
– покупець фільтрує каталог товарів і обирає потрібний;
– покупець оформлює замовлення, яке зберігається до бази даних;
– покупцю висвічується підтвердження збереження його замовлення.

Ця діаграма демонструє взаємодію покупця з системою і моделює один з можливих сценаріїв використання вебзастосунку.

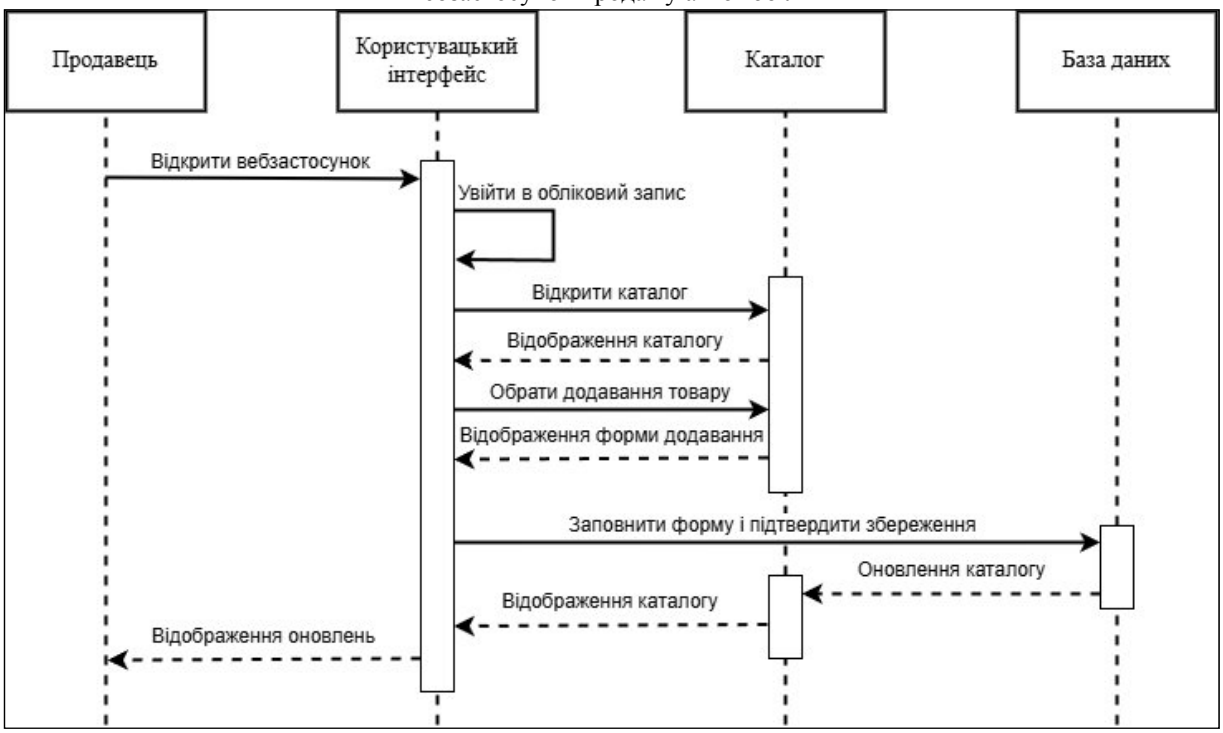

Рисунок 3.5 – Додавання нового товару до каталогу

Дана діаграма зображає процес додавання нового товару до каталогу через заповнення відповідної форми:

– продавець відкриває вебзастосунок;
– входить в свій обліковий запис і відкриває каталог;
– система відображає каталог з правами на керування товарами;
– продавець обирає додати новий товар;
– продавець заповнює форму і підтверджує збереження до бази даних;
– продавцю висвічується оновлений каталог товарів.

Ця діаграма демонструє взаємодію продавця з системою відображаючи можливість продавця керувати товарами.

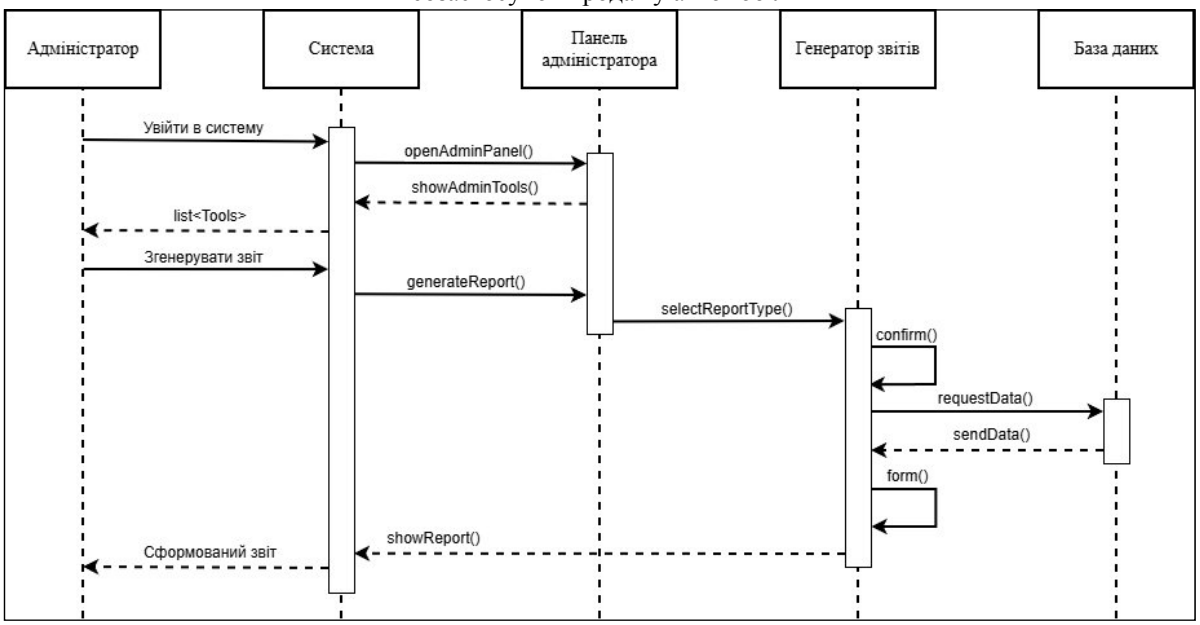

Рисунок 3.6 – Формування звіту

Діаграма демонструє процес формування аналітичного звіту для адміністратора:

– адміністратор входить в систему;
– відкриває панель адміністратора;
– адміністратор обирає функцію «Згенерувати звіт» та тип звіту;
– генератор звітів звертається до бази даних для отримання даних;
– формує звіт і повертає його адміністратору.

Ця діаграма показує, як адміністратор взаємодіє з панеллю адміністратора та модулем генерації звітів. Представлені діаграми послідовності допомагають детально зобразити сценарії роботи системи для користувачів з різними правами доступу та визначити роботу компонентів у вебзастосунку.

Діаграма класів – це одна з головних структурних діаграм. Вона візуалізує архітектуру системи, відображаючи класи разом з їхніми атрибутами і методами та складні зв’язки між ними [25]. Вона дає змогу описати, які класи існують у системі, які властивості та поведінку вони мають, та яким чином класи взаємодіють між собою.

Діаграма класів використовується для:

– проєктування архітектури програмного забезпечення на початку розробки;

– полегшення взаємодії і комунікації між розробниками, тестувальниками та аналітиками;

– документування коду, допомагаючи програмістам швидко зрозуміти логіку системи.

Кожен клас на діаграмі відображається у вигляді прямокутника, який розділено на три частини:

– назва класу;

– атрибути якими володіє клас, які характеризують стан об’єкта;

– методи або функції, які може виконувати клас і які описують поведінку об’єкта.

Зв’язки між класами демонструють логіку взаємодії між об’єктами. Разом все це дозволяє чітко визначити предметну область, повторне використання коду, зробити програмне забезпечення масштабованим та розділити відповідальність між різними частинами системи (рис. 3.7).

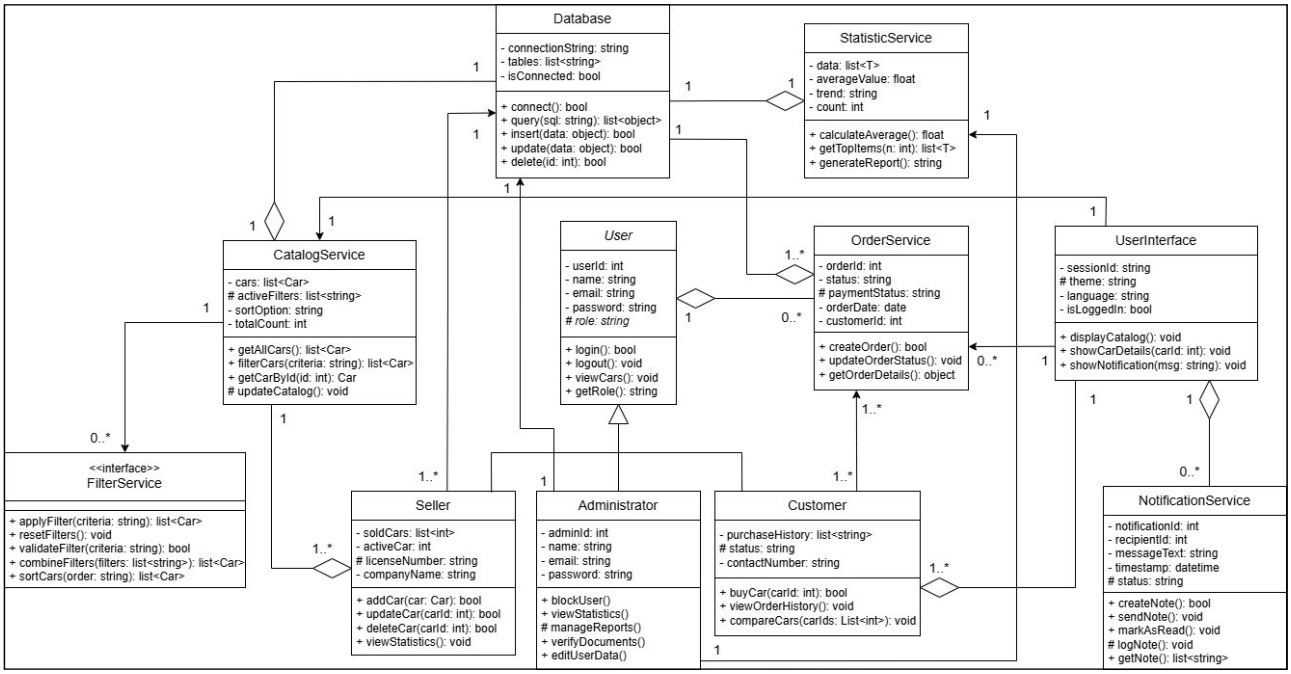

Рисунок 3.7 – Діаграма класів

На діаграмі зображено основні класи системи, які забезпечують правильне функціонування вебзастосунку продажу автомобілів. Кожен з наведених класів

відіграє важливу роль у системі та відповідає певному елементу технічної реалізації програмного забезпечення.

Користувачами системи виступають покупець (customer), продавець (seller) та адміністратор (administrator). Клас покупець представляє користувача вебзастосунку, зберігає дані про нього та містить методи для перегляду авто, додавання авто до таблиці порівняннь та оформлення замовлень. Клас продавець представляє офіційну особу від виробника авто, надає можливість керувати товарами, працювати з каталогом авто та редагувати інформацію про товари виробника якого ця особа представляє. Клас адміністратор реалізує функціональність адміністратора, а саме надає доступ до панелі адміністратора, дозволяє проводити аналіз даних та дає можливість генерувати звіти.

За з’єднання з базою даних відповідає клас Database. Він відкриває сесію зв’язку з базою даних та безпечно закриває її після завершення роботи, надає методи для безпечної відправки команд (читання, запис, оновлення, видалення), забезпечує цілісність даних, об’єднуючи кілька операцій в одну транзакцію та відкочує зміни, якщо сталася помилка.

Також є класи які представляють певні сервіси для вдалого користування вебзастосунком. Одними з таких класів є:

– CatalogService – клас який відповідає за каталог автомобілів. Він працює з класом який відповідає за з’єднання до бази даних і отримує інформацію про автомобілі для демонстрації її на сторінці вебзастосунку.

– FilterService – клас який працює разом з CatalogService і відповідає за фільтрацію авто в каталозі;

– OrderService – клас який відповізає за оформлення замовлень від користувачів.

Діаграма пакетів – це структурна діаграма, яка демонструє організацію елементів системи та залежності між ними. Вона використовується для розбиття великих та складних проєктів на менші логічні групи для полегшення розробки та супроводу програмного забезпечення. Пакетом називають контейнер для елементів системи таких як класи та інтерфейси.

Діаграми пакетів розроблюються для:

– структурування. Це допомагає розділити велику систему на логічні модулі;

– візуалізації залежностей. Це показує які модулі взаємодіють, або залежать один від одного;

– спрощення супроводу і масштабування. Це дозволяє виявити циклічні залежності, які ускладнюють тестування або оновлення програми.

На діаграмі пакетів відображають прямокутники з назвами пакетів між якими показують залежності. Вони демонструють взаємодію, або залежність одного пакету від іншого (рис. 3.8).

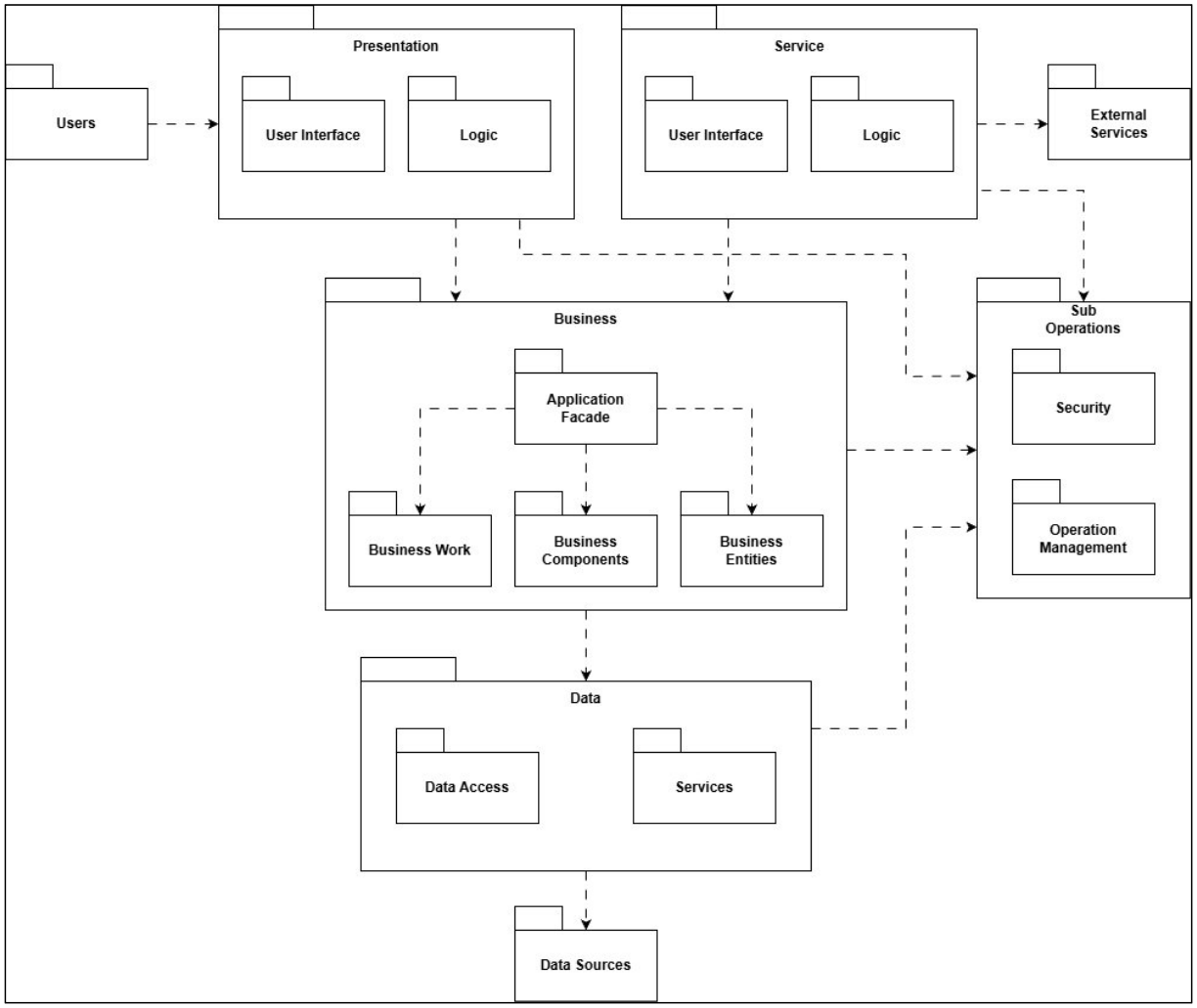

Рисунок 3.8 – Діаграма пакетів

Діаграма пакетів відображає логічну структуру вебзастосунку продажу автомобілів. В якості основних пакетів представлені: Presentation, Service, Business

та Data. Такий поділ відповідає тришаровій архітектурі та забезпечує чітке розмежування відповідальності між компонентами системи.

Пакет Presentation відповідає за рівень представлення та забезпечує взаємодію користувача із системою. До його складу входять сторінки інтерфейсу та контролери, що реагують на дії користувача і передають запити в бізнес-логіку. Пакет Service відповідає за надання сервісів для користувача, наприклад сповіщення вебзастосунку чи використання зовнішніх сервісів таких як: GoogleMaps чи Email.

Попередні два пакети залежать від пакета Business, оскільки сторінки, контролери та сервіси використовують бізнес-логіку. Пакет Business містить логіку предметної області та основні сутності системи. В ньому зосереджені класи, що реалізують функціональні можливості вебзастосунку.

Пакет Business залежить від пакета Data, оскільки логіка працює із базою даних. Цей пакет надає можливість працювати зі збереженими даними, зокрема збереження, пошук та оновлення інформації. Пакет Data реалізує доступ до бази даних, забезпечує зберігання інформації та включає допоміжні компоненти для роботи з даними.

Висновки до розділу 3

У третьому розділі виконано проєктування вебзастосунку продажу автомобілів з урахуванням сучасних вимог. Детально розглянуто архітектуру системи, сформовано моделі, що охоплюють бізнес-процеси та можливості технічної реалізіції.

У результаті проведеного моделювання розроблено IDEF0-діаграми, які дозволили продемонструвати основні процеси вебзастосунку продажу автомобілів. Побудовані моделі відображають взаємодію користувачів із системою та інформаційні потоки між окремими підсистемами застосунку.

Також розглянуто проєктування структури даних системи. Визначено основні сутності предметної області, серед яких користувачі, каталог товарів та заява на купівлю, також встановлено взаємозв’язки між ними. Це дозволило

сформувати логічну основу для подальшої реалізації бази даних та серверної логіки вебзастосунку.

Побудова UML-діаграм варіантів використання, послідовності, класів та пакетів дала можливість в повному обсязі описати структуру та принцип роботи програмного забезпечення. Використання таких моделей дозволяє візуалізувати поведінку системи, визначити взаємодії між компонентами та сформувати чітке уявлення про функціонування вебзастосунку.

У процесі проєктування обгрунтовано використання клієнт-серверної архітектури та сучасного стеку технологій, до якого входять HTML5, CSS3, Angular, JavaScript, TypeScript, ASP.NET та Microsoft SQL Server. Обрані технології забезпечують модульність, масштабованість, зручність супроводу та ефективну взаємодію між клієнтською і серверною частинами системи. Також розроблено структурну модель програмного забезпечення, яка дозволяє організувати взаємодію між основними компонентами системи, забезпечити обробку даних користувачів та роботу з каталогом автомобілів.

4 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

У даному розділі розглядається реалізація розробленого вебзастосунку продажу автомобілів. На основі результатів проєктування здійснюється втілення програмного продукту за допомогою сучасних вебтехнологій та інструментів розробки.

Основна увага приділяється реалізації клієнтської та серверної частин вебзастосунку, організації взаємодії між компонентами системи, а також роботі з базою даних. Під час розробки використано сучасний стек технологій, що забезпечує створення адаптивного користувацького інтерфейсу, швидку обробку запитів користувачів, збереження інформації та реалізацію бізнес-логіки застосунку.

Також розглядається забезпечення модульності системи, зручність супроводу та масштабованість програмного забезпечення. Разом з цим описуються особливості реалізації основних функціональних можливостей вебзастосунку, таких як робота з каталогом автомобілів, оформлення замовлень, авторизація користувачів та порівняння обраних товарів.

В кінці розділу описуються проведені тестування розробленого вебзастосунку. Наводяться підходи до перевірки функціоналу, тест-кейси для основних бізнеспроцесів та результати тестування які відображають коректність роботи системи.

4.1 Структура вебзастосунку

Структура розробленого вебзастосунку продажу автомобілів побудована за принципом розділення клієнтської та серверної частин. Такий підхід забезпечує незалежність frontend та backend компонентів системи, спрощує супровід програмного забезпечення та дозволяє масштабувати окремі частини вебзастосунку незалежно одна від одної.

Проєкт складається з двох основних частин (рис. 4.1): frontend – клієнтська частина вебзастосунку, яка реалізована за допомогою Angular та backend – серверна частина застосунку, яка реалізована на платформі ASP.NET.

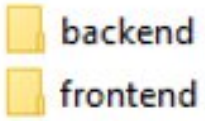

Рисунок 4.1 – Загальна структура проєкту

Frontend частина відповідає за реалізацію користувацького інтерфейсу, взаємодію користувача із системою та відображення самого вебзастосунка. Backend частина забезпечує реалізацію бізнес-логіки, обробку HTTP-запитів, взаємодію з базою даних та роботу REST API. Взаємодія між frontend та backend здійснюється через HTTP-запити із передачею даних у форматі JSON.

Клієнтська частина системи реалізована у вигляді SPA-застосунку з використанням Angular. Такий підхід дозволяє оновлювати окремі частини сторінки без її повного перезавантаження, що покращує швидкість роботи та зручність використання вебзастосунку. Структура Angular-проєкту складається з конфігураційних файлів та директорії *src*. Конфігураційні файли, які знаходяться у кореневому каталозі, відповідають за налаштування середовища розробки, керування залежностями та параметри збірки системи. Директорія *src* є головною папкою клієнтської частини, яка містить основний вихідний код застосунку (рис. 4.2).

Рисунок 4.2 – Структура директорії src

Основними елементами директорії *src* є папки *app* та *assets*. Папка *assets* призначена для зберігання статичних ресурсів вебзастосунку. У ній знаходяться графічні матеріали, зображення автомобілів та інші допоміжні файли, які використовуються під час роботи системи. Папка *app* є директорією яка містить

програмну логіку клієнтської частини. У ній розташовані компоненти, сервіси та моделі даних, які забезпечують роботу вебзастосунку (рис. 4.3).

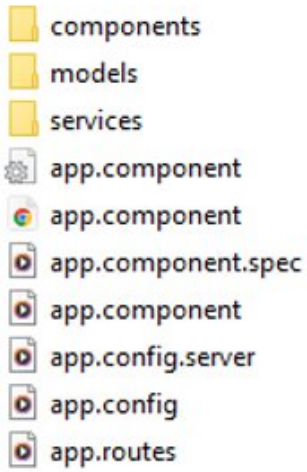

Рисунок 4.3 – Структура директорії app

Головним компонентом виступає *app.component*, який є точкою входу для користувацького інтерфейсу. Він слугує контейнером для всіх інших компонентів і завантажується найпершим під час запуску вебзастосунку.

Файл *app.config* є центральним файлом для налаштування глобальних залежностей (провайдерів) і сервісів. Він відповідає за налаштування HTTP-клієнта, маршрутизації (routing) та інших сервісів.

Файл *app.routes* є основним конфігураційним файлом в Angular, який містить масив маршрутів. Він визначає, який саме компонент має завантажуватися в браузері при переході за певною URL-адресою, і таким чином реалізує односторінкову (SPA) навігацію.

Також у структурі директорії *app* присутні такі папки:

– *models* – зберігає файли моделей даних, які використовуються для типізації інформації у frontend частині вебзастосунку;

– *services* – містить сервіси, які забезпечують взаємодію між компонентами та серверною частиною застосунку;

– *components* – у цій папці знаходяться окремі Angular-компоненти вебзастосунку, які є основною логікою інтерфейсу користувача.

Такий компонентний підхід Angular дозволяє розділити інтерфейс на незалежні функціональні частини, що спрощує повторне використання коду, тестування та подальший супровід програмного забезпечення.

Серверна частина вебзастосунку реалізована за допомогою ASP.NET. Вона забезпечує обробку HTTP-запитів, роботу REST API, реалізацію бізнес-логіки та взаємодію з базою даних (рис. 4.4).

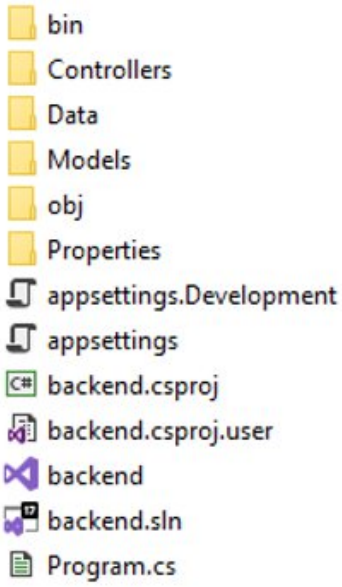

Рисунок 4.4 – Структура backend частини

У цій директорії розміщені конфігураційні файли, які містять інформацію про підключені бібліотеки та залежності, необхідні для роботи серверної частини вебзастосунку, налаштування підключення до бази даних, параметри середовища розробки та інші службові налаштування серверної частини.

Файл *Program.cs* є основним файлом запуску ASP.NET-застосунку. Він відповідає за конфігурацію серверної частини, налаштування сервісів, маршрутизації, підключення контролерів, політик безпеки та запуск вебсерверу.

Папки *bin* та *obj* є службовими директоріями ASP.NET-проєкту. У них зберігаються скомпільовані файли, тимчасові дані та результати збірки застосунку. Папка *Properties* містить службові параметри конфігурації ASP.NET-застосунку та налаштування запуску проєкту у середовищі розробки.

Папка *Controllers* містить API-контролери, які забезпечують взаємодію frontend частини із сервером через HTTP-запити. Кожен контролер відповідає за окрему функціональну частину системи, таку як роботу з автомобілями, замовленнями, користувачами та іншими даними.

Папка *Models* містить моделі даних предметної області. Вони використовуються для опису структури сутностей системи та взаємодії з базою даних через Entity Framework.

Папка *Data* містить класи для роботи з базою даних. Одним із таких класів є *AppDBContext*, який забезпечує підключення до бази даних, налаштування таблиць та виконання операцій збереження і отримання даних.

Така структура backend частини дозволяє розділити відповідальність між окремими компонентами системи, полегшує супровід програмного забезпечення та спрощує подальше масштабування вебзастосунку.

4.2 Структура бази даних

База даних – це організоване сховище взаємопов’язаних даних, яка є одним із головних компонентів вебзастосунку. Вона забезпечує зберігання, організацію та доступ до всієї інформації про користувачів, автомобілі, замовлення та інші сутності системи. Для збереження інформації вебзастосунку продажу автомобілів обрано систему керування базами даних Microsoft SQL Server. Ключовими перевагами якої є високий рівень безпеки, масштабованість, продуктивність обробки даних, а також підтримка механізмів доступності та резервування даних.

У розробленому вебзастосунку продажу автомобілів база даних містить вісім таблиць, кожна з яких відповідає за зберігання та обробку даних певних сутностей у системі (рис. 4.5). Усі таблиці бази даних використовують кодування UTF-8, яке є стандартом для операційних систем і підтримує коректне відображення символів будь-якої мови світу.

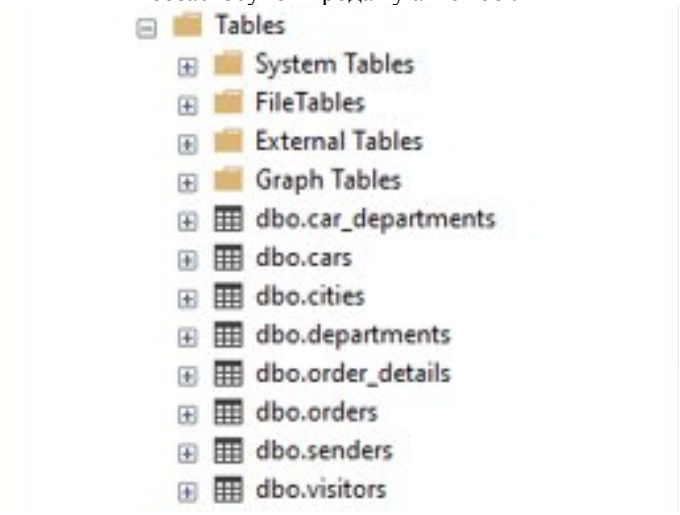

Рисунок 4.5 – Таблиці бази даних

Опис таблиць:

– car_departments (автомобілі у відділеннях) – зберігає інформацію для відстеження наявності кожного автомобіля у різних відділеннях;

– cars (автомобілі) – містить інформацію про усі авто з каталогу включаючи ціну та характеристики;

– cities (міста) – таблиця у якій зберігаються назви міст у яких знаходяться відділення магазину;

– departments (відділення) – зберігає назви відділень магазину;

– order_details (деталі замовлень) – таблиця, яка зберігає додаткову інформацію про замовлення для їх легшого опрацювання;

– orders (замовлення) – містить інформацію про оформлені замовлення користувачів вебзастосунку;

– senders (відправники) – таблиця у якій зберігаються країни у яких виробляють автомобілі, які представлені в каталозі;

– visitors (відвідувачі) – зберігає інформацію про усіх користувачів вебзастосунку, їх логіни та паролі.

Спроєктована структура бази даних забезпечує роботу головних функцій вебзастосунку, серед яких: керування товарами, робота з користувачами та оформлення замовлень. Використання такої структури дозволяє забезпечити

надійне зберігання даних, безпеку, масштабованість та можливість подальшого масштабування.

Лістинг створення таблиці cars

```
CREATE TABLE cars (
    code INT PRIMARY KEY,
    name VARCHAR(500) NOT NULL,
    condition VARCHAR(100) NOT NULL,
    max_speed INT NOT NULL,
    engine VARCHAR(500) NOT NULL,
    capacity DECIMAL(2,1) NOT NULL,
    power INT NOT NULL,
    torque INT NOT NULL,
    drive_unit VARCHAR(500) NOT NULL,
    transmission VARCHAR(500) NOT NULL,
    description TEXT NOT NULL,
    price INT NOT NULL,
    main_image_path VARCHAR(1000) NOT NULL,
    image1_path VARCHAR(1000) NOT NULL,
    image2_path VARCHAR(1000) NOT NULL,
    image3_path VARCHAR(1000) NOT NULL,
    sender_code INT NOT NULL,
    CONSTRAINT FK_Cars_Senders
        FOREIGN KEY (sender_code)
        REFERENCES senders(code)
);
```

Наведений SQL-запит використовується для створення таблиці *cars*. Первинний ключ *code* забезпечує унікальніть кожного автомобіля, а зовнішній ключ *sender_code* реалізує зв’язок з таблицею *senders*, у якій зберігається інформація про країну виробника.

4.3 Керівництво користувача

Для забезпечення зручного використання вебзастосунку продажу автомобілів розроблено керівництво користувача. У ньому описані основні можливості і функції системи, порядок взаємодії користувача з інтерфейсом, а також особливості користування вебзастосунком.

При відкритті вебзастосунку користувач потрапляє на головну сторінку “Home” (рис. 4.6).

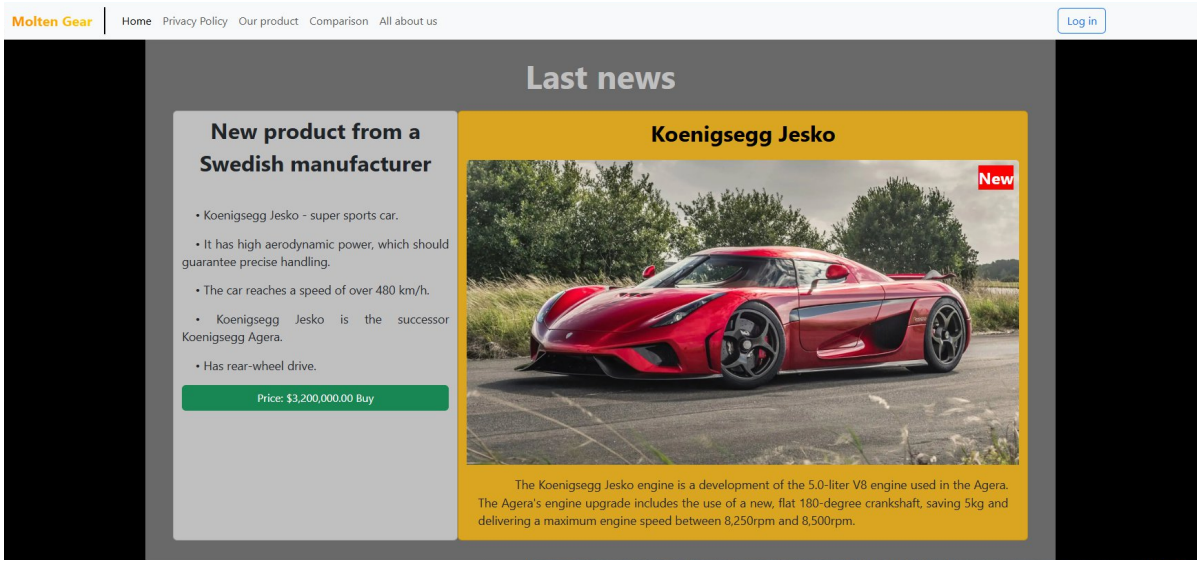

Рисунок 4.6 – Сторінка “Home”

Зверху сторінки розташована навігаційна панель, це один з найважливіших елементів інтерфейсу, який впливає на користувацький досвід та ефективність сайту. Цей елемент забезпечує зручне та швидке переміщення користувачів між різними сторінками та розділами вебзастосунку.

У правій частині навігаційної панелі є кнопка “Log in”. Вона потрібна для реєстрації (якщо користувач вперше зайшов до вебзастосунку) або авторизації користувачів у системі через заповнення форми (рис. 4.7).

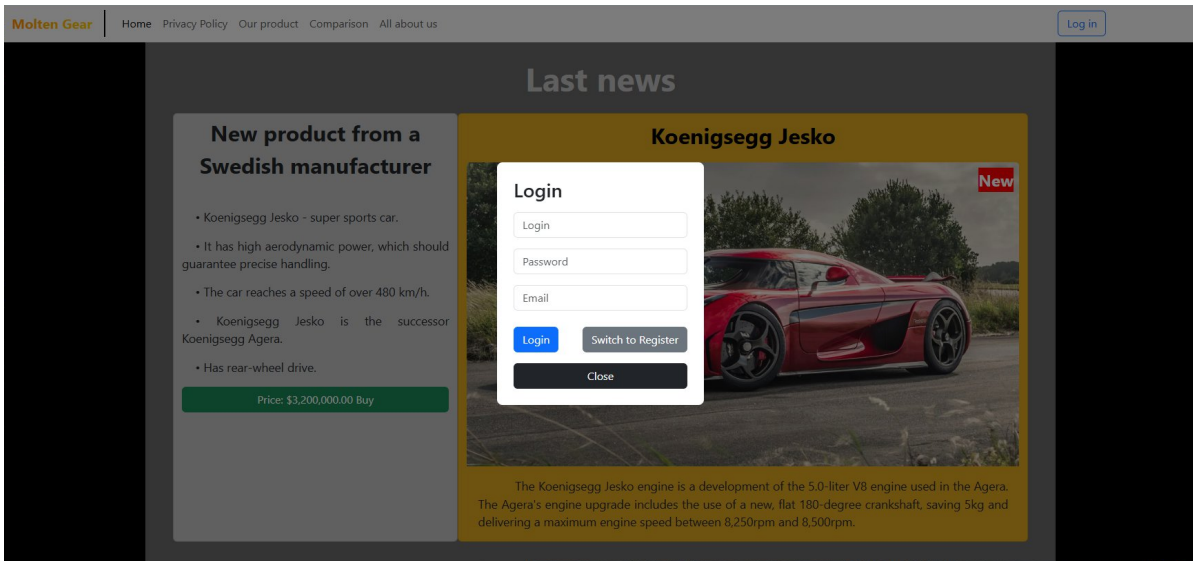

Рисунок 4.7 – Форма реєстрації/авторизації

Для входу до облікового запису користувачу потрібно ввести login, пароль та електронну пошту. Після натискання кнопки “Login”, відбудеться перевірка введених даних з тими, що зберігаються у базі даних і якщо введено коректні дані, користувач увійде до свого облікового запису (рис. 4.8).

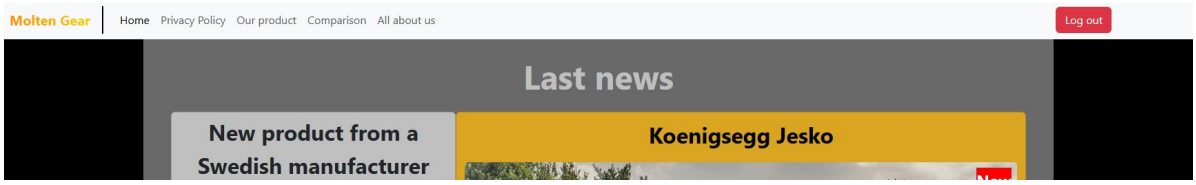

Рисунок 4.8 – Вхід до облікового запису

Головна сторінка “Home”

Знаходячись на сторінці “Home” користувач має змогу ознайомитись з останніми новинами даного вебзастосунку, які включають нові товари, знижки, акції та вигідні пропозиції (рис. 4.9).

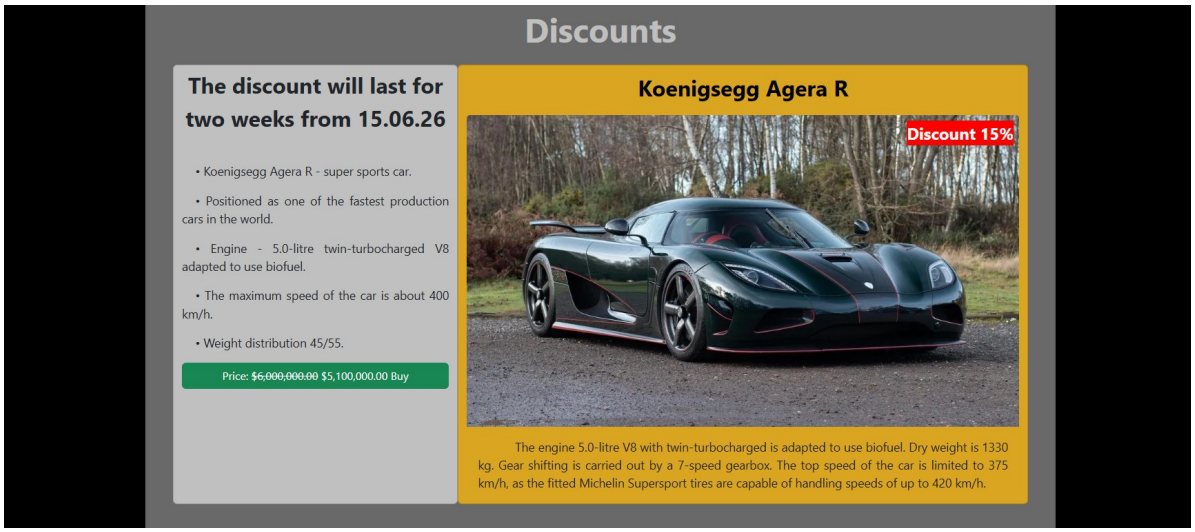

Рисунок 4.9 – Знижки на товари

Якщо під час використання вебзастосунку у користувача виникнуть проблеми у роботі системи, або виникнуть питання, то для зворотнього зв’язку передбачена контактна інформація, яка розташована в нижні частині сторінки (рис. 4.10).

Рисунок 4.10 – Контактна інформація

Сторінка “All about us”

Повертаючись до навігаційної панелі, користувач може перейти до сторінки “All about us”, де розповідається про те чому краще обирати співпрацю з цією компанією, її переваги та послуги які пропонують клієнтам (рис. 4.11 – 4.12).

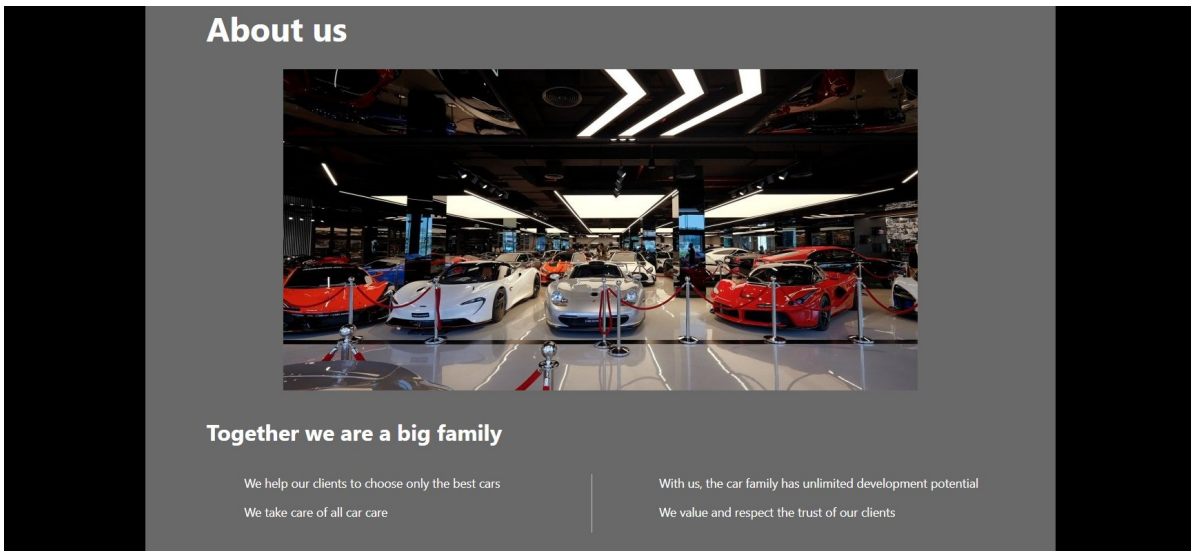

Рисунок 4.11 – Сторінка “All about us”

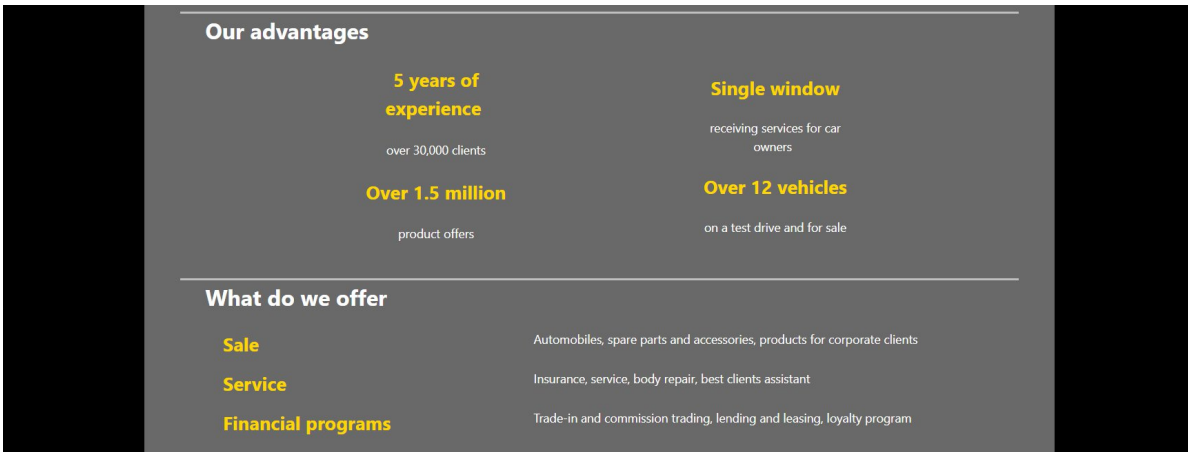

Рисунок 4.12 – Сторінка “All about us” (продовження)

Сторінка “Our product”

Якщо через навігаційну панель натиснути на кнопку “Our product”, користувач перейде до сторінки де відображується каталог автомобілів. Каталог представлений у вигляді карток для кожного автомобіля з його назвою, коротким описом та можливістю додати до порівняння (рис. 4.13).

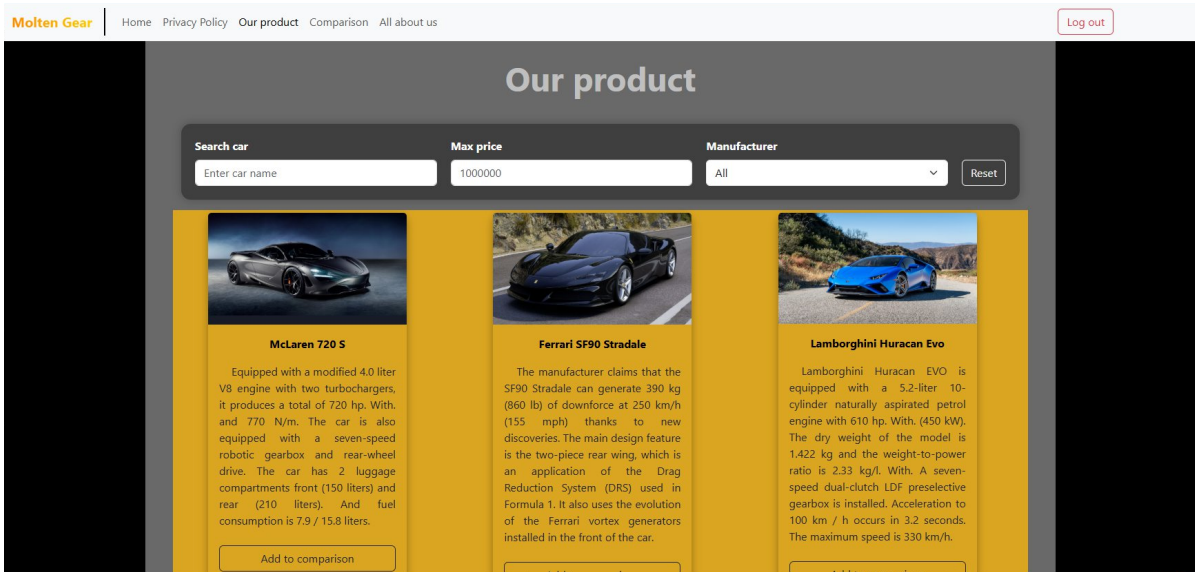

Рисунок 4.13 – Сторінка “Our product”

Для спрощення пошуку потрібного авто, зверху цієї сторінки передбачена фільтрація. Вона забезпечує швидке звуження пошуку серед великої кількості товарів до кількох найбільш підходящих. Це економить час покупця та покращує користувацький досвід.

Сторінка “Comparison”

Під час перегляду каталогу авто користувач може додати обрані авто до таблиці порівнянь. Для цього потрібно натиснути на кнопку “Add to comparison”, яка знаходиться внизу кожної картки авто, та обрати “Continue shopping” для продовження вибору авто, або “Go to comparison” для того щоб перейти до порівняння обраних автомобілів (рис. 4.14 – 4.15).

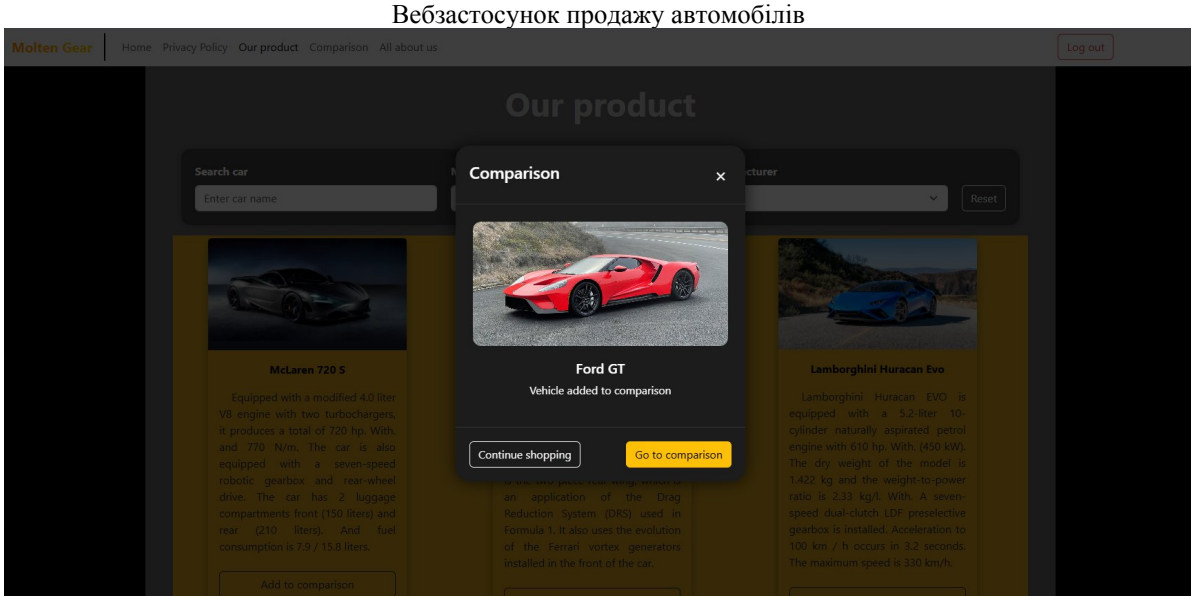

Рисунок 4.14 – Додавання авто до порівняння

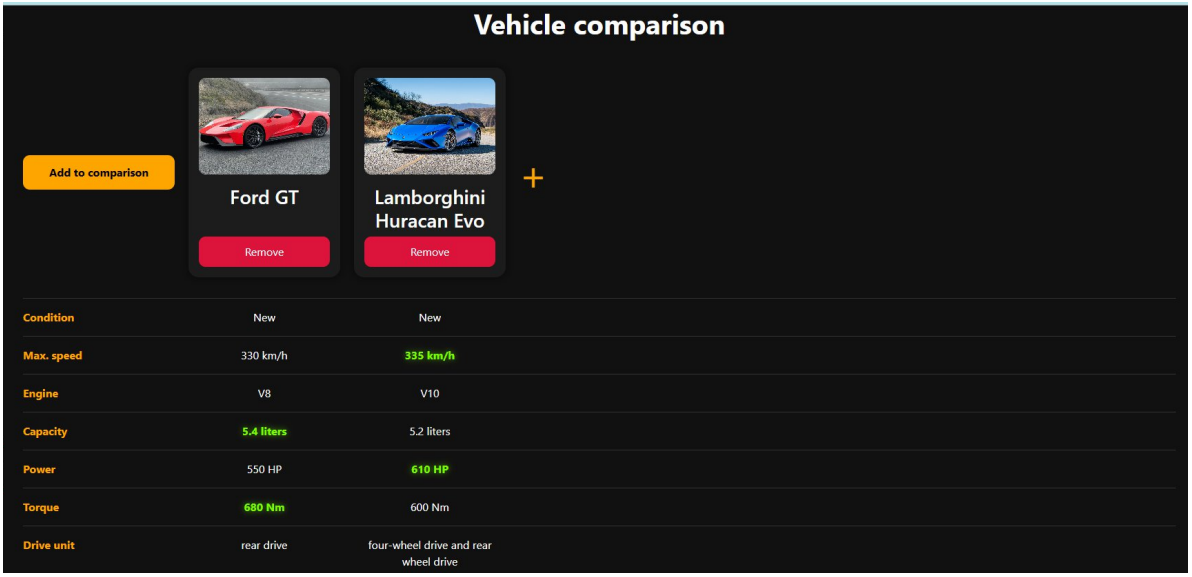

Рисунок 4.15 – Таблиця порівнянь

У цій таблиці порівнюються детальні характеристики кожного авто, які були додані до порівняння. Кращі характеристики підсвічуються зеленим кольором. Також у таблиці передбачено додавання нових авто, з каталогу, до списку порівнюваних та видалення тих які вже додані.

Сторінка детальних характеристик

Повернувшись до каталогу, якщо на картці натиснути на назву авто, користувач перейде до сторінки з детальною інформацією про обраний автомобіль. На цій сторінці представлено карусель фото авто для повного огляду з різних сторін, його детальні характеристики та ціна (рис. 4.16 – 4.17).

Рисунок 4.16 – Карусель фото

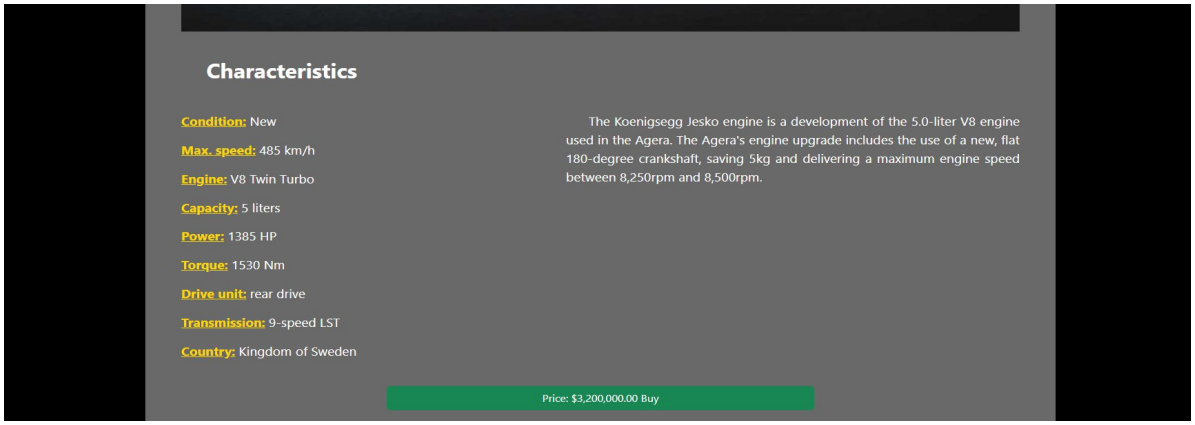

Рисунок 4.17 – Детальні характеристики обраного авто

Якщо натиснути на кнопку “Buy”, користувач перейде на сторінку оформлення замовлення (рис. 4.18).

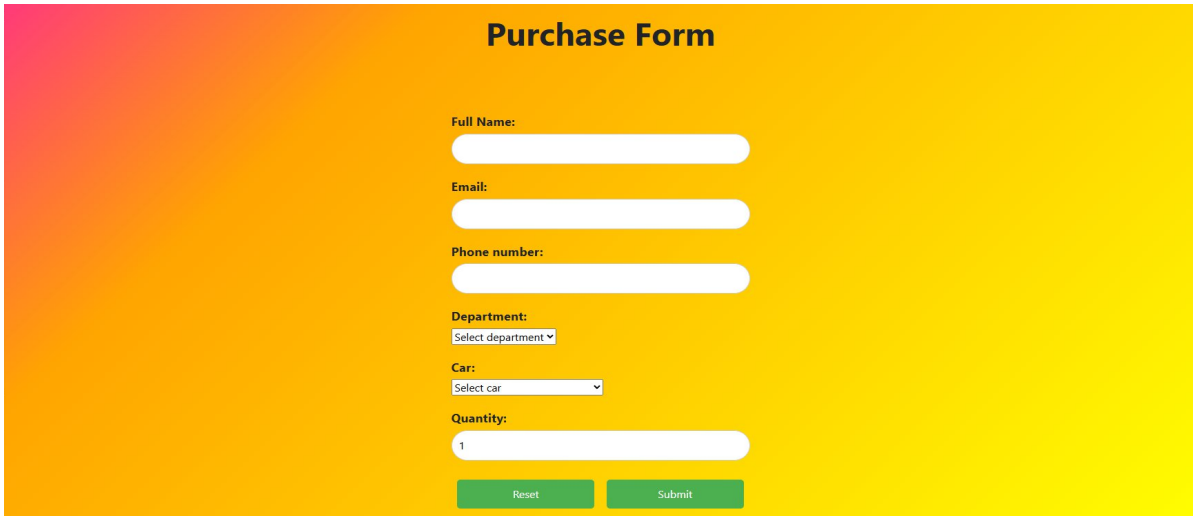

Рисунок 4.18 – Форма оформлення замовлення

Для оформлення замовлення користувачу потрібно ввести особисті дані, такі як ім’я, адресу електроної пошти та номер телефону, а також вказати відділення у якому забирати авто, назву самого авто та кількість. Після коректного заповнення форми замовлення потрібно натиснути кнопку “Submit” для збереження замовлення у базі даних і його подальшого опрацювання.

Для адміністраторів та продавців

Для користувачів з розширеними правами доступу (адміністратор та продавець) на сторінці каталогу автомобілів передбачена можливість керування товарами. Вони можуть додавати нове авто до каталогу, змінювати інформацію про існуючі та видаляти зайві (рис. 4.19 – 4.20).

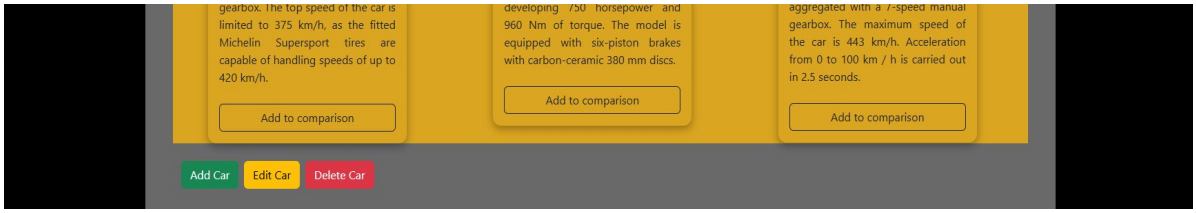

Рисунок 4.19 – Кнопки керування товарами

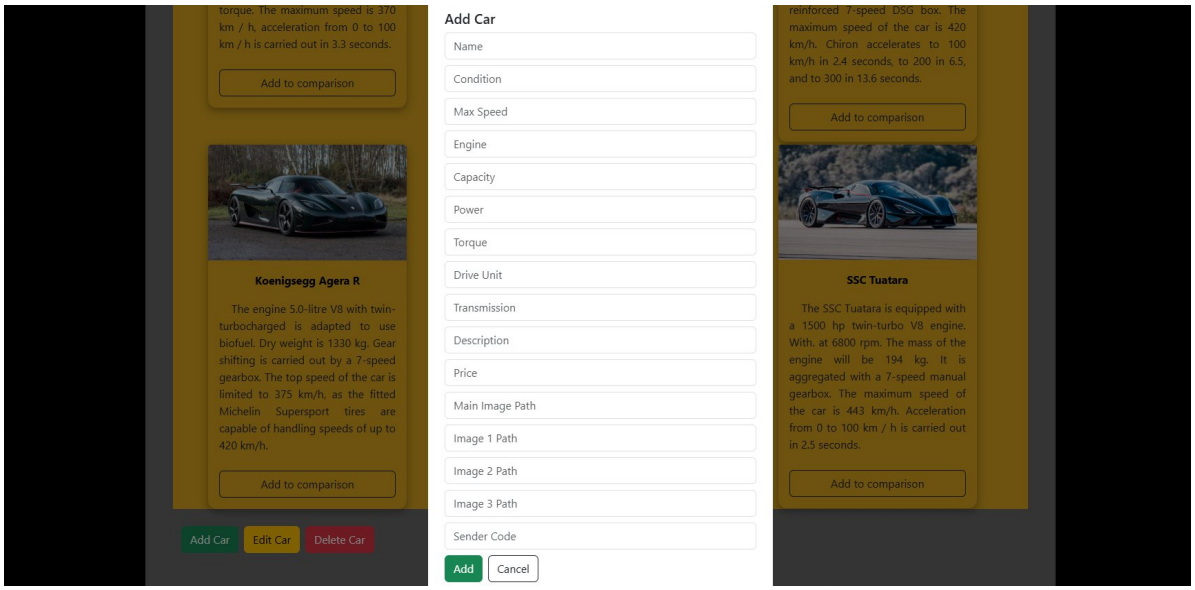

Рисунок 4.20 – Форма додавання нового авто до каталогу

Таким чином, розроблений вебзастосунок забезпечує користувачам зручний доступ до функцій перегляду каталогу автомобілів, порівняння характеристик, оформлення замовлень та взаємодії з системою через вебінтерфейс. Для

адміністраторів і продавців передбачено окремі можливості керування товарами та наповнення каталогу.

4.4 Тестування вебзастосунку

Після реалізації функціоналу вебзастосунку проведено тестування системи з метою перевірки коректності роботи основних модулів, взаємодії клієнтської та серверної частин, а також стабільності роботи вебзастосунку загалом. Проводилось ручне функціональне тестування основних модулів вебзастосунку. Перевірено сценарії роботи для кінцевого користувача (реєстрація/авторизація, фільтрація, додавання автомобілів до порівняння, коректність порівняння, оформлення замовлення), та для адміністратора і продавця (додавання, редагування та видалення автомобілів з каталогу, перегляд замовлень, керування користувачами).

Таблиця 4.1 – Тестування функціоналу вебзастосунку

№	Назва тестування	Кроки виконання	Очікуваний результат	Примітка
1	Реєстрація користувача	Заповнити форму реєстрації	Введені дані збережені до бази даних, користувач потрапив до облікового запису	Виконано успішно
2	Авторизація користувача	Заповнити форму авторизації	Дані перевіряються з тими, що збережені в базі даних, якщо введені коректні дані – користувач входить до облікового запису, інакше показується помилка	Виконано успішно
3	Фільтрація каталогу	Застосувати фільтри	У каталозі показуються автомобілі, які відповідають застосованим фільтрам	Виконано успішно
4	Пошук автомобіля	Ввести назву автомобіля в пошук	У каталозі показуються автомобілі, які відповідають запиту з пошуку	Виконано успішно

Продовження таблиці 4.1

5	Додавання автомобілів до порівняння	Натиснути кнопку “Add to comparison”	Обраний автомобіль з’являється в таблиці порівнянь разом з фото та характкристиками	Виконано успішно
6	Перевірка порівняння	Додати декілька авто до порівняння	Відображуються всі характеристики кожного автомобіля, кращі характеристики пісвічуються зеленим	Виконано успішно
7	Перевірка збереження порівняння	Додати декілька авто до порівняння та оновити сторінку	Додані автомобілі до таблиці порівнянь зберігаються	Виконано успішно
8	Оформлення замовлення	Заповнити та підтвердити відправку замовлення	Перевірка коректності введених даних, якщо введені коректні дані, користувачу відправляється повідомлення про збереження замовлення, інакше помилка про некоректні дані	Виконано успішно
9	Додавання нового автомобіля до каталогу	Заповнити форму додавання нового авто	Перевірка введених даних на відповідність типам даних у рядках бази даних, якщо введені коректні дані – новий автомобіль з’являється в каталозі, інакше помилка додавання	Виконано успішно
10	Зміна інформації про автомобіль	Заповнити форму новими даними	Перевірка введених даних на відповідність типам даних у рядках бази даних, якщо введені коректні дані – інформація про автомобіль змінюється, інакше помилка зміни даних	Виконано успішно

Кінець таблиці 4.1

11	Видалення автомобіля з каталогу	Ввести код автомобіля до форми видалення	Відкриється модальне вікно для підтвердження, або відхилення запиту на видалення автомобіля, якщо відхилення – каталог залишається без змін, якщо підтвердження – оновлений каталог без видаленого авто	Виконано успішно
12	Вихід із системи	Натиснути кнопку “Log out”	Відбудеться вихід з облікового запису	Виконано успішно

У результаті проведених тестів підтверджено коректну роботу основних функцій вебзастосунку. Під час тестування перевірено працездатність механізмів авторизації користувачів, фільтрації та пошуку автомобілів, роботи системи порівняння, оформлення замовлень, а також функцій адміністрування каталогу. Отримані результати свідчать про стабільну взаємодію клієнтської та серверної частин системи, коректну роботу з базою даних та показують, що розроблений вебзастосунок може використовуватися в реальних умовах.

Висновки до розділу 4

У четвертому розділі розглянуто процес реалізації вебзастосунку продажу автомобілів. Описано структуру клієнтської та серверної частин системи, особливості організації бази даних та взаємодії між компонентами застосунку.

Також наведено керівництво користувача із демонстрацією основних функціональних можливостей вебзастосунку та проведено тестування розробленого програмного забезпечення. Результати тестування підтвердили коректність роботи системи, стабільність основних модулів та відповідність реалізованого функціоналу поставленим вимогам. Розроблений вебзастосунок забезпечує зручну взаємодію користувачів із каталогом автомобілів, підтримує оформлення замовлень та надає інструменти адміністрування системи.

ВИСНОВКИ

В процесі виконання кваліфікаційної бакалаврської роботи проведено повний цикл аналізу, проєктування та реалізації вебзастосунку продажу автомобілів. У процесі роботи досліджено предметну область електронної комерції, визначено основні вимоги до програмного забезпечення та проаналізовано сучасні підходи до побудови клієнт-серверних вебзастосунків.

На основі проведеного аналізу для розробки вебзастосунку використано стек технологій Angular, ASP.NET та Microsoft SQL Server, що дозволило створити масштабований та зручний у використанні програмний продукт. Обрані технології забезпечили реалізацію адаптивного користувацького інтерфейсу, серверної логіки, збереження даних та стабільної взаємодії між компонентами системи.

У роботі виконано проєктування структури вебзастосунку та бази даних, побудовано функціональну архітектуру системи, визначено основні бізнес-процеси та реалізовано взаємодію між клієнтською і серверною частинами через REST API. Використання компонентного підходу Angular та модульної структури ASP.NET дозволило забезпечити зручність супроводу, повторне використання коду та можливість подальшого масштабування вебзастосунку.

У процесі реалізації програмного продукту виконано налаштування бази даних, створення основних програмних модулів та інтеграцію всіх компонентів системи. Також проведено тестування основних функціональних можливостей вебзастосунку. Результати тестування підтвердили коректність роботи системи, стабільність взаємодії між frontend та backend частинами, а також відповідність реалізованого функціоналу поставленим вимогам.

Виконання кваліфікаційної роботи дозволило закріпити практичні навички у сфері сучасної веброзробки, роботи з базами даних, тестування програмного забезпечення, та організації структури програмних проєктів. Таким чином, досягнуто поставленої мети, а розроблений вебзастосунок продажу автомобілів може бути використаний як основа для подальшого розвитку та вдосконалення систем електронної комерції у сфері продажу автомобілів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Automotive Insights. S&P Global. URL: https://www.spglobal.com/automotive-insights/en/blogs/2026/01/five-predictions-2026-automotive-industry-outlook (accessed: 21.04.2026).

2. Pikh I., Merenych Y. Semantic models for web application design. Computer Systems and Information Technologies, 2025, (2), 20–26. DOI: 10.31891/csit-2025-2-2.

3. Nissan Україна: офіційний сайт. URL: https://www.nissan.ua/ (дата звернення: 22.04.2026).

4. Renault Україна: офіційний сайт. URL: https://www.renault.ua/ (дата звернення: 22.04.2026).

5. Toyota VIDI: офіційний дилер. URL: https://toyota-ua.com/ua/ (дата звернення: 22.04.2026).

6. RST.ua: продаж автомобілів. URL: https://m.rst.ua/ukr/ (дата звернення: 23.04.2026).

7. AUTO.RIA: продаж автомобілів. URL: https://auto.ria.com/uk/ (дата звернення: 23.04.2026).

8. Клієнт-серверна архітектура. QATestLab training ctnter. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 25.04.2026).

9. Blinowski G., Ojdowska A., Przybyłek A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation // IEEE Access, 2022. Vol. 10, pp. 20357-20374. DOI: 10.1109/ACCESS.2022.3152803.

10. Yakovenko V., Ulianovska Y., Yakovenko T. Analysing features of e-commerce systems architecture // Economics of enterprises. Macroeconomics. Economic Cybernetics: Reports on Research Projects, 2022. Vol. 1 No. 4(63). DOI: 10.15587/2706-5448.2022.253932.

11. Що таке мікросервісна архітектура. COLOBRIDGE. URL: https://blog.colobridge.net/uk/2024/01/what-is-microservices-architecture-ua/ (дата звернення: 25.04.2026).

12. Fedorchuk A., Usata O., Nakonechna O. Series: Engineering science and architecture // Web design and web programming in the modern internet world, 2023. Vol. 6 No. 180. DOI: 10.33042/2522-1809-2023-6-180-12-20.

13. HTML5. HTML Living Standard. URL: https://html.spec.whatwg.org/multipage/ (accessed: 25.04.2026).

14. Difference between HTML, CSS and JavaScript. Medium. URL: https://medium.com/@Bharat2044/difference-between-html-css-and-javascript-51a977ac3ed1 (accessed: 25.04.2026).

15. Неділько О., Сачук В. Порівняння JavaScript-фреймворків React, Angular і Vue.js у дослідженні продуктивності та масштабованості веб-додатків. Herald of Khmelnytskyi National University, 2025. Technical Sciences, 359(6.2), 231-234. DOI: 10.31891/2307-5732-2025-359-103.

16. Пех П., Янковський Б. Особливості створення веб-додатків з використанням C# ASP.NET Core MVC. Комп'ютерно-інтегровані технології: освіта, наука, виробництво, 2025, (59), 253-257. DOI: 10.36910/6775-2524-0560-2024-57-32.

17. Воротнікова З. Огляд баз даних при розробці програмного забезпечення для різних операційних систем. Вісник Приазовського Державного Технічного Університету, 2025. Серія: Технічні науки, (51), 19–31. DOI: 10.31498/2225-6733.51.2025.344596.

18. Getting started with Git. Medium. URL: https://medium.com/@sri.ashish91/getting-started-with-git-42829d31255d (accessed: 25.04.2026).

19. Козуб Г., Сурма Ю., Артеменко О. Роль вебкомпонентів у побудові сучасних інтерфейсів: переваги та обмеження. Вісник Херсонського національного технічного університету, 2025. Том 2 № 2(93). DOI: 10.35546/kntu2078-4481.2025.2.2.21.

20. Дзюрбан Е., Яшина О. Метод оцінки об'єктно-орієнтованих програмних систем на основі аналізу зміни вимог до програмної системи. Herald

of Khmelnytskyi National University, 2022. Technical Sciences, 315 № 6(1), 77-81. DOI: 10.31891/2307-5732-2022-315-6-77-81.

21. Angular. URL: https://angular.dev/reference/configs/file-structure (accessed: 27.04.2026).

22. ASP.NET Web APIs. Microsoft. URL: https://dotnet.microsoft.com/en-us/apps/aspnet/apis (accessed: 27.04.2026).

23. Use case діаграми. Hi-news. URL: https://hi-news.pp.ua/kompyuteri/8924-use-case-dagrama-prikladi-vikoristannya.html (дата звернення: 28.04.2026).

24. Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 29.04.2026).

25. Діаграма класів. ДУІКТ. URL: https://duikt.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 30.04.2026).

ДОДАТОК А

Лістинг коду сторінки каталогу товарів

HTML частина

```
<div class="background">
  <div class="container">
    <div class="header">
      <p>Our product</p>
    </div>
    <div class="filters-wrapper">
      <div class="filters-container">
        <div class="filter-item">
          <label class="form-label">
            Search car
          </label>
          <input
            type="text"
            class="form-control"
            [(ngModel)]="searchText"
            (ngModelChange)="applyFilters()"
            placeholder="Enter car name">
        </div>
        <div class="filter-item">
          <label class="form-label">
            Max price
          </label>
          <input
            type="number"
            class="form-control"
            [(ngModel)]="filterPrice"
            (ngModelChange)="applyFilters()"
            placeholder="1000000">
        </div>
        <div class="filter-item">
          <label class="form-label">
            Manufacturer
          </label>
          <select
            class="form-select"
            [(ngModel)]="filterSender"
            (ngModelChange)="applyFilters()">
            <option value="">
              All
            </option>
            <option value="Bugatti">
              Bugatti
            </option>
            <option value="Ferrari">
```

```
                Ferrari
              </option>
              <option value="Ford">
                Ford
              </option>
              <option value="Koenigsegg">
                Koenigsegg
              </option>
              <option value="Lamborghini">
                Lamborghini
              </option>
              <option value="McLaren">
                McLaren
              </option>
              <option value="Pagani">
                Pagani
              </option>
              <option value="SSC">
                SSC
              </option>
              <option value="W Motors">
                W Motors
              </option>
            </select>
          </div>
          <div class="filter-buttons">
            <button
              class="btn btn-outline-light"
              (click)="resetFilters()">
              Reset
            </button>
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-md-4" *ngFor="let car of cars">
          <div class="card col-md-2 offset-md-1" style="width: 18rem;">
            <img [src]="car.main_image_path" class="card-img-top" alt="Car">
            <div class="card-body">
              <p class="up-text"><a class="look"
[routerLink]="getRouterLink(car.name)">{{ car.name }}</a></p>
              <p class="card-text">{{ car.description }}</p>
              <button class="btn btn-outline-dark w-100 mt-2" (click)="addToComparison(car)">Add
to comparison</button>
            </div>
          </div>
        </div>
      </div>
      <div class="no-results" *ngIf="cars.length === 0">
        <h3>No matching vehicles found</h3>
```

```
    <p>Try changing the search or filter settings.</p>
  </div>
  <div *ngIf="isAdmin" class="admin-controls my-4">
    <button class="btn btn-success me-2" (click)="openAddModal()">Add Car</button>
    <button class="btn btn-warning me-2" (click)="openEditModal()">Edit Car</button>
    <button class="btn btn-danger me-2" (click)="openDeleteModal()">Delete Car</button>
  </div>
  <div *ngIf="showAddModal" class="modal-backdrop">
    <div class="modal-dialog">
      <div class="modal-content p-4">
        <h5>Add Car</h5>
        <form (ngSubmit)="addCarHandler()" #addForm="ngForm">
          <input [(ngModel)]="addCar.name" name="name" required placeholder="Name"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.condition" name="condition" placeholder="Condition"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.max_speed" name="max_speed" type="number"
placeholder="Max Speed" class="form-control mb-2" />
          <input [(ngModel)]="addCar.engine" name="engine" placeholder="Engine" class="form-
control mb-2" />
          <input [(ngModel)]="addCar.capacity" name="capacity" placeholder="Capacity"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.power" name="power" placeholder="Power" class="form-
control mb-2" />
          <input [(ngModel)]="addCar.torque" name="torque" placeholder="Torque" class="form-
control mb-2" />
          <input [(ngModel)]="addCar.drive_unit" name="drive_unit" placeholder="Drive Unit"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.transmission" name="transmission"
placeholder="Transmission" class="form-control mb-2" />
          <input [(ngModel)]="addCar.description" name="description" placeholder="Description"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.price" name="price" type="number" placeholder="Price"
class="form-control mb-2" />
          <input [(ngModel)]="addCar.main_image_path" name="main_image_path"
placeholder="Main Image Path" class="form-control mb-2" />
          <input [(ngModel)]="addCar.image1_path" name="image1_path" placeholder="Image 1
Path" class="form-control mb-2" />
          <input [(ngModel)]="addCar.image2_path" name="image2_path" placeholder="Image 2
Path" class="form-control mb-2" />
          <input [(ngModel)]="addCar.image3_path" name="image3_path" placeholder="Image 3
Path" class="form-control mb-2" />
          <input [(ngModel)]="addCar.sender_code" name="sender_code" type="number"
placeholder="Sender Code" class="form-control mb-2" />
          <button type="submit" class="btn btn-success">Add</button>
          <button type="button" class="btn btn-outline-dark ms-2"
(click)="closeModal()">Cancel</button>
        </form>
      </div>
    </div>
```

```
    </div>
    <div *ngIf="showEditModal" class="modal-backdrop">
      <div class="modal-dialog">
        <div class="modal-content p-4">
          <h5>Edit Car</h5>
          <form (ngSubmit)="editCarHandler()" #editForm="ngForm">
            <input [(ngModel)]="editCar.code" name="code" type="number" required
placeholder="Car Code" class="form-control mb-2" />
            <input [(ngModel)]="editCar.name" name="name" placeholder="Name" class="form-
control mb-2" />
            <input [(ngModel)]="editCar.condition" name="condition" placeholder="Condition"
class="form-control mb-2" />
            <input [(ngModel)]="editCar.max_speed" name="max_speed" type="number"
placeholder="Max Speed" class="form-control mb-2" />
            <input [(ngModel)]="editCar.engine" name="engine" placeholder="Engine" class="form-
control mb-2" />
            <input [(ngModel)]="editCar.capacity" name="capacity" placeholder="Capacity"
class="form-control mb-2" />
            <input [(ngModel)]="editCar.power" name="power" placeholder="Power" class="form-
control mb-2" />
            <input [(ngModel)]="editCar.torque" name="torque" placeholder="Torque" class="form-
control mb-2" />
            <input [(ngModel)]="editCar.drive_unit" name="drive_unit" placeholder="Drive Unit"
class="form-control mb-2" />
            <input [(ngModel)]="editCar.transmission" name="transmission"
placeholder="Transmission" class="form-control mb-2" />
            <input [(ngModel)]="editCar.description" name="description" placeholder="Description"
class="form-control mb-2" />
            <input [(ngModel)]="editCar.price" name="price" type="number" placeholder="Price"
class="form-control mb-2" />
            <input [(ngModel)]="editCar.main_image_path" name="main_image_path"
placeholder="Main Image Path" class="form-control mb-2" />
            <input [(ngModel)]="editCar.image1_path" name="image1_path" placeholder="Image 1
Path" class="form-control mb-2" />
            <input [(ngModel)]="editCar.image2_path" name="image2_path" placeholder="Image 2
Path" class="form-control mb-2" />
            <input [(ngModel)]="editCar.image3_path" name="image3_path" placeholder="Image 3
Path" class="form-control mb-2" />
            <input [(ngModel)]="editCar.sender_code" name="sender_code" type="number"
placeholder="Sender Code" class="form-control mb-2" />
            <button type="submit" class="btn btn-warning">Edit</button>
            <button type="button" class="btn btn-outline-dark ms-2"
(click)="closeModal()">Cancel</button>
          </form>
        </div>
      </div>
    </div>
    <div *ngIf="showDeleteModal" class="modal-backdrop">
      <div class="modal-dialog">
        <div class="modal-content p-4">
```

```
            <h5>Delete Car</h5>
            <form (ngSubmit)="deleteCarHandler()" #deleteForm="ngForm">
              <input [(ngModel)]="deleteCar.code" name="code" type="number" required
placeholder="Car Code" class="form-control mb-2" />
              <button type="submit" class="btn btn-danger">Delete</button>
              <button type="button" class="btn btn-outline-dark ms-2"
(click)="closeModal()">Cancel</button>
            </form>
          </div>
        </div>
      </div><br>
      <div class="header1">
        <p>Interesting facts</p>
      </div>
      <div class="accordion" id="accordionExample">
        <div class="accordion-item">
          <h2 class="accordion-header">
            <button class="accordion-button" type="button" data-bs-toggle="collapse" data-bs-
target="#collapseOne" aria-expanded="true" aria-controls="collapseOne">
              The fastest car in the world
            </button>
          </h2>
          <div id="collapseOne" class="accordion-collapse collapse show" data-bs-
parent="#accordionExample">
            <div class="accordion-body">
              The fastest car in the world at the moment is the SSC Tuatara, a hypercar that achieved
532.93 km/h in one of the races, but the arithmetic mean of 508.73 km/h is considered the official
record.
            </div>
          </div>
        </div>
        <div class="accordion-item">
          <h2 class="accordion-header">
            <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-
bs-target="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo">
              The most expensive supercar in the world
            </button>
          </h2>
          <div id="collapseTwo" class="accordion-collapse collapse" data-bs-
parent="#accordionExample">
            <div class="accordion-body">
              The most expensive car ever sold is a 1962 Ferrari 250 GTO that sold at RM Sotheby's in
2018 for $48.4 million.
            </div>
          </div>
        </div>
        <div class="accordion-item">
          <h2 class="accordion-header">
            <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-
bs-target="#collapseThree" aria-expanded="false" aria-controls="collapseThree">
```

```
                One of the best crossovers
              </button>
            </h2>
            <div id="collapseThree" class="accordion-collapse collapse" data-bs-
parent="#accordionExample">
              <div class="accordion-body">
                The Lamborghini Urus is the first SUV from the Italian car manufacturer Lamborghini.
The car was presented on December 4, 2017.
              </div>
            </div>
          </div>
        </div>
        <div class="header2">
          <p>Contact Information</p>
        </div>
        <div class="coninf">
          <div class="phone">
            <p class="footer">Phone numbers</p>
            <p>+380563842717</p>
            <p>+810657438256</p>
            <p>+330212592069</p>
          </div>
          <div class="supcenadd">
            <p class="footer">Support center address</p>
            <p>Ukraine, Kyiv, Saksaganskogo street 49</p>
            <p>Japan, Tokyo, Omotesando Street 5</p>
            <p>French, Paris, Rivoli Street 77</p>
          </div>
          <div class="e-mail">
            <p class="footer">E-mail</p>
            <p>Engine777&#64;gmail.com</p>
            <p>SuppEng&#64;gmail.com</p>
            <p>SupportCenter&#64;gmail.com</p>
          </div>
        </div>
        <div
          *ngIf="showComparisonModal"
          class="modal-backdrop-custom">
          <div class="comparison-modal">
            <div class="modal-header-custom">
              <h4>Comparison</h4>
              <button
                class="close-button"
                (click)="closeComparisonModal()">
                ×
              </button>
            </div>
            <div class="modal-body-custom">
              <img
                *ngIf="selectedComparisonCar"
```

```
            [src]="selectedComparisonCar.main_image_path"
            class="comparison-modal-image">
          <h5>
            {{ selectedComparisonCar?.name }}
          </h5>
          <p>
            {{ comparisonMessage }}
          </p>
        </div>
        <div class="modal-footer-custom">
          <button
            class="btn btn-outline-light"
            (click)="closeComparisonModal()">
            Continue shopping
          </button>
          <button
            class="btn btn-warning"
            (click)="goToComparison()">
            Go to comparison
          </button>
        </div>
      </div>
    </div>
  </div>
</div>
```

TypeScript частина

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';
import { OurProductService, OurProductCar } from '../../services/our-product.service';
import { ComparisonService } from '../../services/comparison.service';
import { FormsModule } from '@angular/forms';
import { Router } from '@angular/router';

@Component({
  selector: 'app-our-product',
  standalone: true,
  templateUrl: './our-product.component.html',
  styleUrls: ['./our-product.component.css'],
  imports: [CommonModule, RouterModule, FormsModule]
})
export class OurProductComponent implements OnInit {
  cars: OurProductCar[] = [];
  isAdmin = false;
  addCar: Partial<OurProductCar> = {};
  editCar: Partial<OurProductCar> = { code: undefined };
  deleteCar: { code: number | null } = { code: null };

  showAddModal = false;
  showEditModal = false;
```

```
  showDeleteModal = false;
  filterPrice: number | null = null;
  filterSender: string = '';
  searchText: string = '';
  showComparisonModal = false;
  selectedComparisonCar: OurProductCar | null = null;
  comparisonMessage = '';

  constructor(private OurProductService: OurProductService, private comparisonService:
ComparisonService, private router: Router) {}

  ngOnInit() {
    const username = sessionStorage.getItem('username');
    this.isAdmin = username === 'Admin';
    this.loadCars();
  }
  loadCars() {
    this.OurProductService.getCars().subscribe(data => {
      this.cars = data;
    });
  }
  applyFilters() {
    this.OurProductService.getFilteredCars(
      this.filterPrice || undefined,
      this.filterSender || undefined,
      this.searchText || undefined
    ).subscribe(data => {
      this.cars = data;
    });
  }
  resetFilters() {
    this.filterPrice = null;
    this.filterSender = '';
    this.searchText = '';
    this.loadCars();
  }
  addToComparison(car: OurProductCar): void {
    const added = this.comparisonService.addCar(car);
    if (!added) {
      const exists = this.comparisonService
        .getCars()
        .find(c => c.code === car.code);
      if (exists) {
        this.comparisonMessage = 'This vehicle is already in comparison';
        this.selectedComparisonCar = car;
        this.showComparisonModal = true;
      }
      return;
    }
    this.comparisonMessage = 'Vehicle added to comparison';
```

```
  this.selectedComparisonCar = car;
  this.showComparisonModal = true;
}

closeComparisonModal(): void {
  this.showComparisonModal = false;
}

goToComparison(): void {
  this.showComparisonModal = false;
  this.router.navigate(['/comparison']);
}

getRouterLink(name: string): string {
  return '/' + name.toLowerCase().replace(/ /g, '-') + '-look';
}

openAddModal() {
  this.addCar = {};
  this.showAddModal = true;
}
openEditModal() {
  this.editCar = {};
  this.showEditModal = true;
}
openDeleteModal() {
  this.deleteCar = { code: null };
  this.showDeleteModal = true;
}
closeModal() {
  this.showAddModal = false;
  this.showEditModal = false;
  this.showDeleteModal = false;
}

addCarHandler() {
  if (!this.addCar.name || !this.addCar.price) {
    alert('Please fill in all required fields.');
    return;
  }

  this.OurProductService.addCar(this.addCar as OurProductCar).subscribe({
    next: () => {
      alert('Car added successfully');
      this.addCar = {};
      this.closeModal();
      this.loadCars();
    },
    error: err => {
      console.error('Failed to add car', err);
```

```
      alert('Failed to add car');
    }
  });
 }

 editCarHandler() {
  if (!this.editCar.code) {
   alert('Please enter the car code to edit');
   return;
  }

  this.OurProductService.updateCar(this.editCar.code, this.editCar as OurProductCar).subscribe({
   next: () => {
    alert('Car updated successfully');
    this.editCar = {};
    this.closeModal();
    this.loadCars();
   },
   error: err => {
    console.error('Failed to update car', err);
    alert('Failed to update car');
   }
  });
 }

 deleteCarHandler() {
  const code = this.deleteCar.code;
  if (code === null) {
   alert('Please enter the car code to delete');
   return;
  }

  if (!confirm('Are you sure you want to delete this car?')) return;

  this.OurProductService.deleteCar(code).subscribe({
   next: () => {
    alert('Car deleted successfully');
    this.closeModal();
    this.loadCars();
   },
   error: err => {
    console.error('Failed to delete car', err);
    alert('Failed to delete car');
   }
  });
 }
}
```

ДОДАТОК Б

Лістинг коду сторінки порівнянь

HTML частина

```
<div class="comparison-page">
  <h1 class="title">
    Vehicle comparison
  </h1>
  <div *ngIf="cars.length === 0" class="empty-message">
    No cars added to comparison
  </div>
  <div *ngIf="cars.length > 0">
    <div class="comparison-header">
      <div class="feature-column">
        <button class="add-button" (click)="goToProducts()">
          Add to comparison
        </button>
      </div>
      <div
        class="car-column"
        *ngFor="let car of cars">
        <img
          [src]="car.main_image_path"
          class="car-image">
        <h3>{{ car.name }}</h3>
        <button
          class="remove-button"
          (click)="removeCar(car.code)">
          Remove
        </button>
      </div>
      <div class="plus-column" (click)="goToProducts()">
        +
      </div>
    </div>
    <div class="comparison-row">
      <div class="feature-name">
        Condition
      </div>
      <div
        class="feature-value"
        *ngFor="let car of cars">
        {{ car.condition }}
      </div>
    </div>
    <div class="comparison-row">
      <div class="feature-name">
        Max. speed
```

```
    </div>
    <div
      class="feature-value"
      [class.best-value]="isBestMaxSpeed(car.max_speed)"
      *ngFor="let car of cars">
      {{ car.max_speed }} km/h
    </div>
  </div>
  <div class="comparison-row">
    <div class="feature-name">
      Engine
    </div>
    <div
      class="feature-value"
      *ngFor="let car of cars">
      {{ car.engine }}
    </div>
  </div>
  <div class="comparison-row">
    <div class="feature-name">
      Capacity
    </div>
    <div
      class="feature-value"
      [class.best-value]="isBestCapacity(car.capacity)"
      *ngFor="let car of cars">
      {{ car.capacity }} liters
    </div>
  </div>
  <div class="comparison-row">
    <div class="feature-name">
      Power
    </div>
    <div
      class="feature-value"
      [class.best-value]="isBestPower(car.power)"
      *ngFor="let car of cars">
      {{ car.power }} HP
    </div>
  </div>
  <div class="comparison-row">
    <div class="feature-name">
      Torque
    </div>
    <div
      class="feature-value"
      [class.best-value]="isBestTorque(car.torque)"
      *ngFor="let car of cars">
      {{ car.torque }} Nm
    </div>
```

```
      </div>
      <div class="comparison-row">
        <div class="feature-name">
          Drive unit
        </div>
        <div
          class="feature-value"
          *ngFor="let car of cars">
          {{ car.drive_unit }}
        </div>
      </div>
      <div class="comparison-row">
        <div class="feature-name">
          Transmission
        </div>
        <div
          class="feature-value"
          *ngFor="let car of cars">
          {{ car.transmission }}
        </div>
      </div>
      <div class="comparison-row">
        <div class="feature-name">
          Price
        </div>
        <div
          class="feature-value"
          *ngFor="let car of cars">
          ${{ car.price }}
        </div>
      </div>
    </div>
</div>
```

CSS частина

```
.comparison-page {
  background: #111;
  min-height: 100vh;
  color: white;
  padding: 30px;
}
.title {
  text-align: center;
  margin-bottom: 40px;
  font-size: 40px;
  font-weight: bold;
}
.comparison-header {
  display: flex;
  align-items: stretch;
  margin-bottom: 30px;
```

```
}

.feature-column {
    width: 220px;
    display: flex;
    align-items: center;
    justify-content: center;
}
.add-button {
    width: 100%;
    height: 50px;
    border: none;
    background: orange;
    color: black;
    font-weight: bold;
    border-radius: 10px;
}
.car-column {
    display: flex;
    flex-direction: column;
    width: 220px;
    background: #1c1c1c;
    margin-left: 20px;
    padding: 15px;
    border-radius: 15px;
    text-align: center;
    box-shadow: 0 0 15px rgba(0,0,0,0.5);
}
.car-image {
    width: 100%;
    height: 140px;
    object-fit: cover;
    border-radius: 10px;
    margin-bottom: 15px;
}
.remove-button {
    width: 100%;
    margin-top: auto;
    border: none;
    background: crimson;
    color: white;
    padding: 10px;
    border-radius: 10px;
}
.plus-column {
    width: 80px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 60px;
```

```
    color: orange;
    cursor: pointer;
}

.comparison-row {
    display: flex;
    border-top: 1px solid #333;
    padding: 15px 0;
}
.feature-name {
    width: 220px;
    color: orange;
    font-weight: bold;
}
.feature-value {
    width: 220px;
    text-align: center;
    margin-left: 20px;
}
.best-value {
    color: #7CFC00;
    font-weight: bold;
    text-shadow: 0 0 10px rgba(124, 252, 0, 0.6);
}
.empty-message {
    text-align: center;
    font-size: 30px;
    margin-top: 100px;
    color: gray;
}
```

TypeScript частина

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ComparisonService } from '../../services/comparison.service';
import { OurProductCar } from '../../services/our-product.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-comparison',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './comparison.component.html',
  styleUrls: ['./comparison.component.css']
})
export class ComparisonComponent implements OnInit {
  cars: OurProductCar[] = [];

  constructor(
    private comparisonService: ComparisonService,
    private router: Router
```

```
  ) {}

  ngOnInit(): void {
    this.loadCars();
  }

  loadCars(): void {
    this.cars = this.comparisonService.getCars();
  }
  removeCar(code: number): void {
    this.comparisonService.removeCar(code);
    this.loadCars();
  }

  goToProducts(): void {
    this.router.navigate(['/our-product']);
  }

  isBestMaxSpeed(value: number): boolean {
    const maxSpeed = Math.max(
      ...this.cars.map(car => car.max_speed)
    );
    return value === maxSpeed;
  }
  isBestCapacity(value: number): boolean {
    const maxCapacity = Math.max(
      ...this.cars.map(car => car.capacity)
    );
    return value === maxCapacity;
  }
  isBestPower(value: number): boolean {
    const maxPower = Math.max(
      ...this.cars.map(car => car.power)
    );
    return value === maxPower;
  }
  isBestTorque(value: number): boolean {
    const maxTorque = Math.max(
      ...this.cars.map(car => car.torque)
    );
    return value === maxTorque;
  }
}
```